Linear-algebraic λ-calculus: higher-order, encodings, and confluence.

Pablo Arrighi*
Université de Grenoble,
46 Ave F. Viallet, 38031 Grenoble Cedex, France.

Gilles Dowek†
École polytechnique and INRIA,
LIX, 91128 Palaiseau Cedex, France.

We introduce a minimal language combining higher-order computation and linear algebra. Roughly, this is nothing else than the λ-calculus together with the possibility to make arbitrary linear combinations of terms $\alpha.\mathbf{t} + \beta.\mathbf{u}$. We describe how to "execute" this language in terms of a few rewrite rules, and justify them through the two fundamental requirements that the language be a language of linear operators, and that it be higher-order. We mention the perspectives of this work in the field of quantum computation, whose circuits we show can be easily encoded in the calculus. Finally we prove the confluence of the calculus, this is our main result.

I. MOTIVATIONS

The objective of this paper is to merge higher-order computation, be it terminating or not, in its simplest and most general form (namely the untyped λ-calculus) together with linear algebra in its simplest and most general form also (we take just an oriented version of the axioms of vectorial spaces). We see this as a platform for various applications, including quantum computation, each of them probably requiring its own type systems. Next we develop the various contexts in which this calculus may bring some decisive advances.

Quantum programming languages. Over the last two decades, the discovery of several great algorithmic results [11, 16, 26] has raised important expectations in the field of quantum computation. Somewhat surprisingly however these results have been expressed in the primitive model of quantum circuits – a situation which is akin to that of classical computation in the 1950s. Over the last few years a number of researchers have sought to develop quantum programming languages as a consequence. Without aiming to be exhaustive and in order to understand where the perspectives of this work come in, it helps to classify these proposals according to "how classical" versus "how quantum" they are [25].

There are two ways a quantum mechanical system may evolve: according to a unitary transformation or under a measurement. The former is often thought of as "purely quantum": it is deterministic and will typically be used to obtain quantum superpositions of base vectors. The latter is probabilistic in the classical sense, and will typically be used to obtain some classical information about a quantum mechanical system, whilst collapsing the system to a mere base vector.

*Electronic address: Pablo.Arrighi@imag.fr
†Electronic address: Gilles.Dowek@polytechnique.edu

Note that these are only typical uses: it is well-known that one can simulate any unitary transformation by series of generalized measures on the one hand, and reduce all measures to a mere projection upon the canonical basis at the end of a computation on the other hand. It remains morally true nonetheless that measurement-based models of quantum computation tend to hide quantum superpositions behind a classical interface, whilst the unitary-based models of quantum computation tend to consider quantum superpositions as legitimate expressions of the language, and sometimes even seek to generalize their effects to control flow.

Therefore one may say that measurement-based models of quantum computation – whether reliant upon teleportation[20], state transfer [22] or more astonishingly graph states [23] – lie on one extreme, as they keep the "quantumness" to a minimum.

A more balanced approach is to allow for both unitary transformations and quantum measurements. Such models can be said to formalize the existing algorithm description methods to a strong extent: they exhibit quantum registers upon which quantum circuits may be applied, together with classical registers and programming structures in order to store measurements results and control the computation [24]. For this reason they are the more practical route to quantum programming. Whilst this juxtaposition of "quantum data, classical control" has appeared ad-hoc and heterogeneous at first, functional-style approaches together with linear type systems [2, 25] have ended up producing elegant quantum programming languages.

Finally we may evacuate measures altogether – leaving them till the end of the computation and outside the formalism. This was the case for instance in [27][28], but here the control structure remained classical. In our view, such a language becomes even more interesting once we have also overcome the need for any additional classical registers and programming structures, and aim to draw the full consequence of quantum mechanics: "quantum

data, quantum control". Quantum Turing Machines [5], for instance, lie on this other extreme, since the entire machine can be in a superposition of base vectors. Unfortunately they are a rather oblivious way to describe an algorithm. Functional-style control structure, on the other hand, seem to merge with quantum evolution descriptions in a unifying manner. The functional language we describe may give rise to a "purely quantum" programming language, i.e. one which has no classical registers, no classical control structure, no measurement and that allows arbitrary quantum superpositions of base vectors – once settled the question of restricting to unitary operators.

General computable linear operators, and the restriction to unitary. In our view, the problem of formulating a simple algebra of higher order computable operators upon infinite dimensional vector spaces was the first challenge that needed to be met, before even aiming to have a physically executable language. In the current state of affairs computability in vector spaces is dealt with with matrices and compositions, and hence restricted to finite-dimensional systems – although this limitation is sometimes circumvented by introducing an extra classical control structure e.g. via the notions of uniform circuits or linear types. The language we provide achieves this goal of a minimal calculus for describing higher-order computable linear operators in a wide sense. Therefore this work may serve as a basis for studying wider notions computability upon abstract vector spaces.

The downside of this generality as far as the previously mentioned application to quantum computation are concerned is that our operators are not restricted to being unitary. A further step towards specializing our language to quantum computation would be to restrict to unitary operators, as required by quantum physics. There may be several ways to do so. A first lead would be to design an a posteriori static analysis that enforces unitarity – exactly like typability is not wired in pure lambda-calculus, but may be enforced a posteriori. A second one would be to require a formal unitarity proof from the programmer. With a term and a unitarity proof, we could derive a more standard representation of the operator, for instance in terms of an universal set of quantum gates [8]. This transformation may be seen as part of a compilation process.

In its current state our language can be seen as a specification language for quantum programs, as it possesses several desirable features of such a language: it allows a high level description of algorithms without any commitment to a particular architecture, it allows the expression of black-box algorithms through the use of higher order functionals, its notation remains close to both linear algebra and functional languages. The game is then be to prove that some quantum program expressed in a standard way (as a composition of universal quantum gates, say) is observationally equivalent to such a specification (a term of our language) under the operational semantics given next.

Type theory, logics, models. In this article linearity is understood in the sense of linear algebra, but a further aim to this research would be to investigate connections with linear λ-calculus, i.e. a calculus which types are formulae of linear logic [15]. In such a λ-calculus, one distinguishes linear resources, which may be neither duplicated nor discarded, from nonlinear resources, whose fate is not subjected to particular restrictions. The linear-algebraic λ-calculus we describe bears strong resemblances with the linear λ-calculus, as well as some crucial, strongly motivated differences. Duplication of a term $\mathbf{t}$ is again treated cautiously, but in a finer way: only terms expressing base vectors can be duplicated, which is compatible with linear algebra. As we shall see, terms of the form $\lambda\mathbf{x}\ \mathbf{u}$ are always base vectors. As a consequence, even when a term $\mathbf{t}$ cannot be duplicated the term $\lambda\mathbf{x}\ \mathbf{t}$ can. Since the term $\lambda\mathbf{x}\ \mathbf{t}$ is a function building the term $\mathbf{t}$, it can be thought of as a description of $\mathbf{t}$. This suggests some connections between the abstraction $\lambda\mathbf{x}$, the ! (bang) operator of linear lambda-calculus and the $'$ (quote) operator that transforms a term into a description of it as used for instance in LISP.

The paper may also be viewed as part of a wave of probabilistic extensions of calculi, e.g. [7, 17]. Type theories for probabilistic extensions of the λ-calculus such as ours or the recent [14] may lead to interesting forms of quantitative logics. The idea of superposing λ-terms is also reminiscent of several other works in λ-calculus, in particular Boudol's parallel λ-calculus [6], Ehrhard and Regnier's differential λ-calculus [13, 29] (although our scalars are not restricted to being naturals or finitely splitting and positive numbers, but may be arbitrary numbers), Dougherty's algebraic extension [12] for normalizing terms of the λ-calculus.

The functions expressed in our language are linear operators upon the space constituted by its terms. It is strongly inspired from [4] where terms clearly form a vector space. However because it is higher-order, as functions may be passed as arguments to other functions, we get forms of infinity coming into the game. Thus, the underlying algebraic structure is not as obvious as in [4]. In this paper we provide the rules for executing the language in a consistent fashion (confluence), but we leave open the precise nature of the model which lies beneath.

Confluence techniques. A standard way to describe how a program is executed is to give a small step operational semantic for it, in the form of a finite set rewrite rules which gradually transform a program into a value. The main theorem proved in this paper is the confluence of our language. What this means is that the order in which those transformations are applied does not affect the end result of the computation. Confluence results are milestones in the study of programming languages and more generally in the theory of rewriting. Our proof uses many of the theoretical tools that have been developed for confluence proofs in a variety of fields (local confluence and Newman's lemma; strong confluence and the Hindley-Rosen lemma) as well as the avatar lemma for parametric

rewriting as introduced in [3]. These are fitted together in an elaborate architecture which may have its own interest whenever one seeks to merge a non-terminating conditional confluent rewrite system together with a a terminating conditional confluent rewrite system.

Outline. Section II presents the designing principles of the language, Section III formally describes the linear-algebraic λ-calculus and its semantics. Section IV shows that the language is expressive enough for classical and quantum computations. These are the more qualitative sections of the paper. We chose to postpone till Section V the proof of the confluence of the calculus, which is more technical. This is our main result.

II. MAIN FEATURES OF THE LANGUAGE

We introduce a minimal language combining higher-order computation and linear algebra, i.e. we extend the λ-calculus with the possibility to make linear combinations of terms $\alpha.\mathbf{t} + \beta.\mathbf{u}$.

Higher-order. In computer science many algorithms fall into the category of "black-box" algorithms. I.e. some mysterious implementation of a function f is provided to us which we call "oracle" – and we wish to evaluate some property of f, after a limited number of queries to its oracle. For instance in the Deutsch-Josza quantum algorithm, f is a function $f : \{\mathbf{false}, \mathbf{true}\}^n \longrightarrow \{\mathbf{false}, \mathbf{true}\}$ which is either constant (i.e. $\exists c \forall x[f(x) = c]$) or balanced (i.e. $|\{x \,\text{such that}\, f(x) = \mathbf{false}\}| = |\{x \,\text{such that}\, f(x) = \mathbf{true}\}|$), whose corresponding oracle is a unitary transformation $U_f : \mathcal{H}_2^{n+1} \longrightarrow \mathcal{H}_2^{n+1}$ such that $U_f : \mathbf{x} \otimes \mathbf{b} \mapsto \mathbf{x} \otimes (\mathbf{b} \oplus f(\mathbf{x}))$, where $\mathcal{H}_2^{n+1}$ stands for a tensor product of $n+1$ two-dimensional Hilbert spaces, $\otimes$ is the tensor product and $\oplus$ just the addition modulo two. Our aim is to determine whether f is constant or balanced, and it turns out we can do so in one single query to its oracle. The algorithm works by applying $H^{\otimes^{n+1}}$ upon $(\mathbf{false}^{\otimes^n} \otimes \mathbf{true})$, then U_f, and then $H^{\otimes^{n+1}}$ again, where $H^{\otimes^{n+1}}$ means applying the Hadamard gate on each of the $n+1$ qubits. It is clear from this example that a desirable feature for a linear-algebraic functional language is to be able to express algorithms as a function of an oracle. E.g. we may want to define

$$\mathbf{Dj}_1 \equiv \lambda\mathbf{x}\,((H \otimes H)\,(\mathbf{x}\,((H \otimes H)\,(\mathbf{false} \otimes \mathbf{true}))))$$

so that $\mathbf{Dj}_1\, U_f$ reduces to $(H \otimes H)\,(U_f\,((H \otimes H)\,(\mathbf{false} \otimes \mathbf{true})))$. More importantly even, one must be able to express algorithms, whether they are "black-box" or not, independent of the size of their input. This is what differentiates programs from fixed-size circuits acting upon finite dimensional vector spaces, and demonstrates the ability to have control flow. The way to achieve this in functional languages involves duplicating basic components of the algorithm an appropriate number of times. E.g. we may want to define some $\mathbf{Dj}$ operator so that $(\mathbf{Dj}\,\mathbf{n})\,U_f$ reduces to the appropriate $(\mathbf{Dj}_n)\,U_f$, where $\mathbf{n}$ is a natural number.

Clearly the languages of matrices and circuits do not offer an elegant presentation for this issue. Higher-order appears to be a desirable feature to have for black-box computing, but also for expressing recursion and for high-level programming.

Copying. We seek to design a λ-calculus, i.e. have the possibility to introduce and abstract upon variables, as a mean to express functions of these variables. In doing so, we must allow functions such as $\lambda\mathbf{x}\,(\mathbf{x} \otimes \mathbf{x})$, which duplicate their argument. This is necessary for expressiveness, for instance in order to obtain the fixed point operator or any other form of iteration/recursion.

Now problems come up when functions such as $\lambda\mathbf{x}\,(\mathbf{x} \otimes \mathbf{x})$ are applied to superpositions (i.e. sums of vectors). Linear-algebra brings a strong constraint: we know that cloning is not allowed, i.e. that the operator which maps any vector ψ onto the vector $\psi \otimes \psi$ is not linear. In quantum computation this impossibility is referred to as the "no-cloning theorem" [30]. Most quantum programming language proposals so far consist in some quantum registers undergoing unitary transforms and measures on the one hand, together with classical registers and programming structures ensuring control flow on the other, precisely in order to avoid such problems. But as we seek to reach beyond this duality and obtain a purely quantum programming language, we need to face it in a different manner.

This problem may be seen as a confluence problem. Faced with the term $(\lambda\mathbf{x}\,(\mathbf{x} \otimes \mathbf{x}))\,(\mathbf{false} + \mathbf{true})$, one could either start by substituting $\mathbf{false} + \mathbf{true}$ for $\mathbf{x}$ and get the normal form $(\mathbf{false} + \mathbf{true}) \otimes (\mathbf{false} + \mathbf{true})$, or start by using the fact that all the functions defined in our language must be linear and get $((\lambda\mathbf{x}\,(\mathbf{x} \otimes \mathbf{x}))\,\mathbf{false}) + ((\lambda\mathbf{x}\,(\mathbf{x} \otimes \mathbf{x}))\,\mathbf{true})$ and finally the normal form $(\mathbf{false} \otimes \mathbf{false}) + (\mathbf{true} \otimes \mathbf{true})$, leading to two different results. More generally, faced with a term of the form $(\lambda\mathbf{x}\,\mathbf{t})\,(\mathbf{u} + \mathbf{v})$, one could either start by substituting $\mathbf{u} + \mathbf{v}$ for $\mathbf{x}$, or start by applying the right-hand-side linearity of the application, breaking the confluence of the calculus. So that operations remain linear, it is clear that we must start by developing over the $+$ first, until we reach a base vector and then apply β-reduction. By base vector we mean a term which does not reduce to a superposition. Therefore we restrict the β-reduction rules to cases where the argument is a base vector, as formalized later.

With this restriction, we say that our language allows *copying* but not *cloning* [2, 4]. It is clear that copying has all the expressiveness required in order to express control flow, since it behaves exactly like the standard β-reduction as long as the argument passed is not in a superposition. This is the appropriate linear extension of the β-reduction, philosophically it comprehends classical computation as a (non-superposed) sub-case of linear-algebraic/quantum computation.

The same applies to erasing: the term $\lambda\mathbf{x}\lambda\mathbf{y}\ \mathbf{x}$ expresses the linear operator mapping the base vector

$\mathbf{b}_i \otimes \mathbf{b}_j$ to $\mathbf{b}_i$. Again this is in contrast with other programming languages where erasing is treated in a particular fashion whether for the purpose of linearity of bound variables or the introduction of quantum measurement.

Higher-order & copying. The main conceptual difficulty when seeking to let our calculus be higher-order is to understand how it combines with this idea of "copying", i.e. duplicating only base vectors. Terms of the form $(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))\,(\lambda\mathbf{x}\,\mathbf{v})$ raise the important question of whether the λ-term $\lambda\mathbf{x}\,\mathbf{v}$ must be considered to be a base vector or not. We will now proceed to examine the different possible ways in which one can approach this problem, so as to justify our choice – the reader can skip the discussion if he wants.

- Consider $\lambda\mathbf{x}\,\mathbf{x}$. This is akin to $\sum_i \mathbf{b}_i \rhd \mathbf{b}_i$, where each $\mathbf{b}_i \rhd \mathbf{b}_i$ would be some projector term such that $(\mathbf{b}_i \rhd \mathbf{b}_i)\mathbf{b}_j \longrightarrow^* \delta_{ij}\mathbf{b}_i$, with $(\mathbf{b}_i)_i$ the computational basis.

  Hence, in this approach, $\lambda\mathbf{x}\,\mathbf{x}$ is not a base vector. The problem we then face is that we no longer know how $(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))\,(\lambda\mathbf{x}\,\mathbf{x})$ reduces. Indeed, in order to favor copy over cloning we must first develop as $\sum_i(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))(\mathbf{b}_i \rhd \mathbf{b}_i) + \ldots$, which we know how to do only if $\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x})$ acts over a finite domain. Since our terms are infinite and countable, such a design option must be abandoned. Moreover this would be rather difficult to program with, for instance applying some black-box algorithm which iterates several times the function passed as argument would scatter this black-box into its component projectors, instead of using it as whole. For instance applying the function $\lambda\mathbf{f}\,((\mathbf{f}\ \mathbf{false}) \otimes (\mathbf{f}\ \mathbf{true}))$ to the identity, would scatter the identity into two projectors $\mathbf{false} \rhd \mathbf{false}$ and $\mathbf{true} \rhd \mathbf{true}$, compute the term $((\mathbf{f}\ \mathbf{false}) \otimes (\mathbf{f}\ \mathbf{true}))$ for each of them, yielding the null vector in both cases and sum the results. We would thus obtain the result $\mathbf{0}$ instead of the expected vector $\mathbf{false} \otimes \mathbf{true}$.

- Consider $\lambda\mathbf{x}\,\mathbf{x}$. This is akin to the string "$\sum_i \mathbf{b}_i \rhd \mathbf{b}_i$", or rather to the classical description of a quantum machine which leaves its input unchanged. Hence $\lambda\mathbf{x}\,\mathbf{x}$ is a base vector. We favor this option. In this setting $(\lambda\mathbf{f}\,((\mathbf{f}\,\mathbf{false}) \otimes (\mathbf{f}\,\mathbf{true})))\,(\lambda\mathbf{x}\,\mathbf{x})$ must reduce into $\mathbf{false} \otimes \mathbf{true}$, but there is another dilemma.

  – Now consider $\lambda\mathbf{x}\,(\mathbf{x} + \mathbf{b})$, with $\mathbf{b}$ a base vector. This is akin to $\lambda\mathbf{x}\,\mathbf{x} + \lambda\mathbf{x}\,\mathbf{b}$ and hence it is not a base vector. As a consequence $(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))\,(\lambda\mathbf{x}\,(\mathbf{x} + \mathbf{b}))$ must reduce into $(\lambda\mathbf{x}\,\mathbf{x}) + \mathbf{b}$, by linearity. In other words we ask that $\lambda\mathbf{x}\,(\mathbf{t} + \mathbf{u}) \longrightarrow \lambda\mathbf{x}\,\mathbf{t} + \lambda\mathbf{x}\,\mathbf{u}$ and $\lambda\mathbf{x}\,\alpha.\mathbf{t} \longrightarrow \alpha.\lambda\mathbf{x}\,\mathbf{t}$, i.e. that the abstraction be a unary linear symbol. In general this means that $\lambda\mathbf{x}\,\mathbf{v}$ may be a superposition state, and must be tested for it. This is a rather difficult task if an application appears in $\mathbf{v}$ for instance, and supposing it may create superpositions out of base vectors. Then the danger is that terms such as $(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))$ cannot reduce further as a consequence, whereas in order to recover the classical λ-calculus and its expressiveness such terms should loop forever. But actually without any extension such a design option makes it impossible to create superpositions out of base vectors. These will always have to preexist somehow in the term, resulting in a lack of expressiveness.

  – Consider $\lambda\mathbf{x}\,(\mathbf{x}+\mathbf{b})$. This is akin to the string "$\sum_i \mathbf{b}_i \rhd (\mathbf{b}_i + \mathbf{b})$", or rather to the classical description of a machine which adds $\mathbf{b}$ to its input. Hence $\lambda\mathbf{x}\,(\mathbf{x} + \mathbf{b})$ is a base vector. As a consequence $(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))\,(\lambda\mathbf{x}\,(\mathbf{x}+\mathbf{b}))$ must reduce into $(\lambda\mathbf{x}\,(\mathbf{x}+\mathbf{b}))+\mathbf{b}$. In other words, we ask that abstractions wear an implicit LISP-like quote operator, i.e. they are classical descriptions of machines performing some operation, and hence they are always base vectors. The term $(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))(\lambda\mathbf{x}\,(\mathbf{x}\,\mathbf{x}))$ loops forever and we recover the classical call by value λ-calculus as a special case. This choice comes at no cost for expressiveness, since terms such as $(\lambda\mathbf{x}\,\mathbf{x}) + (\lambda\mathbf{x}\,\mathbf{b})$ remain allowed, but have a different interpretation from $\lambda\mathbf{x}(\,\mathbf{x}+\mathbf{b})$. They act the same, but are acted upon differently. We favor this option.

Hence we can now define base vectors as being either abstractions (since we have taken the view that these are descriptions of machines performing some operation) or variables (because in the end these will always be substituted for a base vector).

Base (in)dependence. It is clear that there is a notion of privileged basis arising in the calculus, but without us having to a priori choose a canonical basis (e.g. we do not introduce some arbitrary orthonormal basis $\{|i\rangle\}$ all of a sudden – i.e. we have nowhere specified a basis at the first-order level). A posteriori, we could have replaced the above full blown discussion by the more direct argument: - we need to restrict $(\lambda\mathbf{x}\,\mathbf{t})\,\mathbf{u} \longrightarrow \mathbf{t}[\mathbf{u}/\mathbf{x}]$ to "base vectors";
- we want higher-order in the traditional sense $(\lambda\mathbf{x}\,\mathbf{t})\,(\lambda\mathbf{y}\,\mathbf{u}) \longrightarrow \mathbf{t}[\lambda\mathbf{y}\,\mathbf{u}/\mathbf{x}]$;
- therefore abstractions must be the base vectors;
- since variables will only ever be substituted by base vectors, they also are base vectors.
The eventual algebraic consequences of this notion of a privileged basis arising only because of the higher-order level are left as a topic for further investigations. An important intuition is that $(\lambda x\ \mathbf{v})$ is not the vector itself, but its classical description, i.e. the machine constructing it – hence it is acceptable to be able to copy $(\lambda x\ \mathbf{v})$ so long as we cannot clone $\mathbf{v}$. The calculus does exactly this distinction.

Infinities & confluence. It is possible, in our calculus, to

define fixed point operators such as

$$\mathbf{Y} = \lambda \mathbf{y}\left((\lambda \mathbf{x}\,(\mathbf{y} + (\mathbf{x}\,\mathbf{x})))\,(\lambda \mathbf{x}\,(\mathbf{y} + (\mathbf{x}\,\mathbf{x})))\right)$$

Then, if $\mathbf{b}$ is a base state, the term $(\mathbf{Y}\,\mathbf{b})$ reduces to $\mathbf{b}+(\mathbf{Y}\,\mathbf{b})$, i.e. the term reductions generate a computable series of vectors $(n.\mathbf{b} + (\mathbf{Y}\,\mathbf{b}))_n$ whose "norm" grows towards infinity. This was expected in the presence of both fixed points and linear algebra, but the appearance of such infinities entails the appearance of indefinite forms, which we must handle with great caution. Marrying the full power of untyped λ-calculus, including fixed point operators etc., with linear-algebra therefore jeopardizes the confluence of the calculus, unless we introduce some further restrictions.

Example 1 *If we took an unrestricted factorization rule* $\alpha.\mathbf{t}+\beta.\mathbf{t} \longrightarrow (\alpha+\beta).\mathbf{t}$, *then the term* $(\mathbf{Y}\,\mathbf{b})-(\mathbf{Y}\,\mathbf{b})$ *would reduce to* $(1 + (-1)).(\mathbf{Y}\,\mathbf{b})$ *and then* $\mathbf{0}$. *It would also be reduce to* $\mathbf{b} + (\mathbf{Y}\,\mathbf{b}) - (\mathbf{Y}\,\mathbf{b})$ *and then to* $\mathbf{b}$, *breaking the confluence.*

Thus, exactly like in elementary calculus $\infty - \infty$ cannot be simplified to 0, we need to introduce a restriction to the rule allowing to factor $\alpha.\mathbf{t} + \beta.\mathbf{t}$ into $(\alpha + \beta).\mathbf{t}$ to the cases where $\mathbf{t}$ is finite. But what do we mean by finite? Notions of norm in the usual mathematical sense seem difficult to import here. In order to avoid infinities we would like to ask that $\mathbf{t}$ be normalizable, but this is impossible to test in general. Hence, we restrict further this rule to the case where the term $\mathbf{t}$ is normal. It is quite striking to see how this restriction equates the algebraic notion of "being normalized" with the rewriting notion of "being normal". The next two examples show that this indefinite form may pop up in some other, more hidden, ways.

Example 2 *Consider the term* $(\lambda \mathbf{x}\,((\mathbf{x}\,\_) - (\mathbf{x}\,\_)))\,(\lambda \mathbf{y}\,(\mathbf{Y}\,\mathbf{b}))$ *where* $\_$ *is any base vector, for instance* **false**. *If the term* $(\mathbf{x}\,\_) - (\mathbf{x}\,\_)$ *reduced to* $\mathbf{0}$ *then this term would both reduce to* $\mathbf{0}$ *and also to* $(\mathbf{Y}\,\mathbf{b}) - (\mathbf{Y}\,\mathbf{b})$, *breaking confluence.*

Thus, the term $\mathbf{t}$ we wish to factor must also be closed, so that it does not contain any hidden infinity.

Example 3 *If we took an unrestricted rule* $(\mathbf{t}+\mathbf{u})\,\mathbf{v} \longrightarrow (\mathbf{t}\,\mathbf{v}) + (\mathbf{u}\,\mathbf{v})$ *the term* $(\lambda \mathbf{x}\,(\mathbf{x}\,\_) - \lambda \mathbf{x}\,(\mathbf{x}\,\_))\,(\lambda \mathbf{y}\,(\mathbf{Y}\,\mathbf{b}))$ *would reduce to* $(\mathbf{Y}\,\mathbf{b}) - (\mathbf{Y}\,\mathbf{b})$ *and also to* $\mathbf{0}$, *breaking confluence.*

Thus we have to restrict this rule to the case where $\mathbf{t}+\mathbf{u}$ is normal and closed.

Example 4 *If we took an unrestricted rule* $(\alpha.\mathbf{u})\,\mathbf{v} \longrightarrow \alpha.(\mathbf{u}\,\mathbf{v})$ *then the term* $(\alpha.(\mathbf{x}+\mathbf{y}))\,(\mathbf{Y}\,\mathbf{b})$ *would reduce both to* $(\alpha.\mathbf{x} + \alpha.\mathbf{y})\,(\mathbf{Y}\,\mathbf{b})$ *and to* $\alpha.((\mathbf{x} + \mathbf{y})\,(\mathbf{Y}\,\mathbf{b}))$, *breaking confluence due to the previous restriction.*

Thus we have to restrict this rule to the case where $\mathbf{u}$ is normal and closed.

This discussion motivates each of the restrictions $(*)$ – $(****)$ in the rules below. These restrictions are not just a fix: they are a way to formalize vectorial spaces in the presence of limits/infinities. It may come as a surprise, moreover, that we are able to tame these infinities with this small added set of restrictions, and without any need for context-sensitive conditions, as we shall prove in Section V.

III. THE LANGUAGE

We consider a first-order language, called *the language of scalars*, containing at least constants 0 and 1 and binary function symbols + and ×. The *language of vectors* is a two-sorted language, with a sort for vectors and a sort for scalars, described by the following term grammar:

$$\mathbf{t} ::= \quad \mathbf{x} \quad | \quad \lambda \mathbf{x}\,\mathbf{t} \quad | \quad (\mathbf{t}\ \mathbf{t}) \quad | \quad \mathbf{0} \quad | \quad \alpha.\mathbf{t} \quad | \quad \mathbf{t} + \mathbf{t}$$

where α has the sort of scalars.

In this paper we consider only semi-open terms, i.e. terms containing vector variables but no scalar variables. In particular all scalar terms will be closed.

As usual we write $(\mathbf{t}\,\mathbf{u}_1\,...\,\mathbf{u}_n)$ for $(...(\mathbf{t}\,\mathbf{u}_1)\,...\,\mathbf{u}_n)$. Vectors appear in bold.

Definition 1 (The system S – scalar rewrite system) *A scalar rewrite system S is an arbitrary rewrite system defined on scalar terms and such that*

- *S is terminating and confluent on closed terms,*
- *for all closed terms α, β and γ, the pair of terms*
  - $0 + \alpha$ *and* α, $0 \times \alpha$ *and* 0, $1 \times \alpha$ *and* α,
  - $\alpha \times (\beta + \gamma)$ *and* $(\alpha \times \beta) + (\alpha \times \gamma)$,
  - $(\alpha+\beta)+\gamma$ *and* $\alpha+(\beta+\gamma)$, $\alpha+\beta$ *and* $\beta+\alpha$,
  - $(\alpha \times \beta) \times \gamma$ *and* $\alpha \times (\beta \times \gamma)$, $\alpha \times \beta$ *and* $\beta \times \alpha$

  have the same normal forms,
- 0 *and* 1 *are normal terms.*

Examples of scalar rewrite systems for $\mathbb{D}$ and $\mathbb{D}[i, \sqrt{2}]$ are given in [4], where $\mathbb{D}$ is the set of rational numbers whose denominators is a power of two, as this is enough to express quantum computations. The same thing could be done for $\mathbb{Q}$ or any finite extension of $\mathbb{Q}$. Basically the notion of a scalar rewrite systems lists the few basic properties that scalars are usually expected to have: neutral elements, associativity of + etc. The following two definitions are standard for rewriting modulo associativity and commutativity.

Definition 2 (The relation $\longrightarrow_{XAC}$) *We define the relation $=_{AC}$ as the congruence generated by the associativity and commutativity axioms of the symbol +. Let X be a rewrite system, we define the relation $\longrightarrow_{XAC}$ as follows $t \longrightarrow_{XAC} u$ if there exists a term t' such*

that $t =_{AC} t'$, an occurrence p in t', a rewrite rule $l \longrightarrow r$ in X and a substitution σ such that $t'_{|p} = \sigma l$ and $u =_{AC} t'[p \leftarrow \sigma r]$).

Definition 3 (The system L − vector spaces) *We consider a system called L formed with the rules of S and the union of four groups of rules E, F, A and B:*
- Group E – elementary rules
$\mathbf{u} + \mathbf{0} \longrightarrow \mathbf{u}$, $\quad 0.\mathbf{u} \longrightarrow \mathbf{0}$, $\quad 1.\mathbf{u} \longrightarrow \mathbf{u}$, $\quad \alpha.\mathbf{0} \longrightarrow \mathbf{0}$,
$\alpha.(\beta.\mathbf{u}) \longrightarrow (\alpha \times \beta).\mathbf{u}$, $\quad \alpha.(\mathbf{u} + \mathbf{v}) \longrightarrow \alpha.\mathbf{u} + \alpha.\mathbf{v}$
- Group F – factorisation
$\alpha.\mathbf{u} + \beta.\mathbf{u} \longrightarrow (\alpha + \beta).\mathbf{u}$, $\quad \alpha.\mathbf{u} + \mathbf{u} \longrightarrow (\alpha + 1).\mathbf{u}$, $\quad \mathbf{u} + \mathbf{u} \longrightarrow (1 + 1).\mathbf{u} \quad (*)$
- Group A – application
$(\mathbf{u} + \mathbf{v})\ \mathbf{w} \longrightarrow (\mathbf{u}\ \mathbf{w}) + (\mathbf{v}\ \mathbf{w})$, $\quad \mathbf{w}\ (\mathbf{u} + \mathbf{v}) \longrightarrow (\mathbf{w}\ \mathbf{u}) + (\mathbf{w}\ \mathbf{v})$, $\quad (**)$
$(\alpha.\mathbf{u})\ \mathbf{v} \longrightarrow \alpha.(\mathbf{u}\ \mathbf{v})$, $\quad \mathbf{v}\ (\alpha.\mathbf{u}) \longrightarrow \alpha.(\mathbf{v}\ \mathbf{u}) \quad (***)$
$\mathbf{0}\ \mathbf{u} \longrightarrow \mathbf{0}$, $\quad \mathbf{u}\ \mathbf{0} \longrightarrow \mathbf{0}$
- Group B – beta reduction
$(\lambda \mathbf{x}\ \mathbf{t})\ \mathbf{b} \longrightarrow \mathbf{t}[\mathbf{b}/\mathbf{x}] \quad (****)$ *where $+$ is an AC symbol. And:*
$(*)$ the three rules apply only if $\mathbf{u}$ is a closed L-normal term.
$(**)$ the two rules apply only if $\mathbf{u}+\mathbf{v}$ is a closed L-normal term.
$(***)$ the two rules apply only if $\mathbf{u}$ is a closed L-normal term.
$(****)$ the rule apply only when $\mathbf{b}$ is a "base vector" term, i.e. an abstraction or a variable.

Notice that the restriction $(*)$, $(**)$ and $(***)$ are well-defined as the terms to which the restrictions apply are smaller than the left-hand side of the rule.
Notice also that the restrictions are stable by substitution. Hence these conditional rules could be replaced by an infinite number of non conditional rules, i.e. by replacing the restricted variables by all the closed normal terms verifying the conditions.
Finally notice how the rewrite system $R = S \cup E \cup F \cup A$, taken without restrictions, is really just an oriented version of the axioms of vectorial spaces, as is further explained in [3]. Intuitively the restricted systems defines a notion of vectorial space with infinities.

Normal forms. We have explained why abstractions ought to be considered as "base vectors" in our calculus. We have highlighted the presence of non-terminating terms and infinities, which make it impossible to interpret the calculus in your usual vector space structure. The following two results show that terminating closed terms on the other hand can really be viewed as superposition of abstractions.

Proposition 1 *An L-closed normal form that is not a sum, a product by a scalar, or the null vector is an abstraction.*

Proof. By induction over term structure. Let $\mathbf{t}$ be a closed normal term that is not a sum, a product by a scalar, or the null vector. The term $\mathbf{t}$ is not a variable because it is closed, hence it is either an abstraction in which case we are done, or an application. In this case it has the form $(\mathbf{u}\,\mathbf{v}_1\,\ldots\,\mathbf{v}_n)$ where $\mathbf{u}, \mathbf{v}_1, \ldots \mathbf{v}_n$ are normal and closed and n is different from 0. Neither $\mathbf{u}$ nor $\mathbf{v}_1$ is a sum, a product by a scalar, or the null vector since the term being normal we then could apply rules of group A. Thus by induction hypothesis both terms are abstractions. Hence the term is not normal because of rule B.

Proposition 2 (Form of closed normal forms) *A L-closed normal form is either the null vector or of the form*

$$\sum_i \alpha_i.\lambda \mathbf{x}\,\mathbf{t}_i + \sum_i \lambda \mathbf{x}\,\mathbf{u}_i$$

Proof. If the term is not the null vector it can be written as a sum of terms that are neither $\mathbf{0}$ nor sums. We partition these terms in order to group those which are weighted by a scalar and those which are not. Hence we obtain a term of the form

$$\sum \alpha'_i.\mathbf{t'}_i + \sum \mathbf{u'}_i$$

where the terms $\mathbf{u'}_i$ are neither null, nor sums, nor weighted by a scalar. Hence by Proposition 1 they are abstractions. Because the whole term is normal the terms $\mathbf{t'}_i$ are themselves neither null, nor sums, nor weighted by a scalar since we could apply rules of group E. Hence Proposition 1 also applies.

IV. ENCODING CLASSICAL AND QUANTUM COMPUTATION

The restrictions we have placed upon our language are still more permissive than those of the call-by-value λ-calculus, hence any classical computation can be expressed in the linear-algebraic λ-calculus just as it can in can in the call-by-value λ-calculus. For expressing quantum computation we need a specific language of scalars, together with its scalar rewrite system. This bit is not difficult, as was shown in [4]. It then suffices to express the three universal quantum gates $\mathbf{H}, \mathbf{Phase}, \mathbf{Cnot}$, which we will do next.

Encoding booleans. We encode the booleans as the first and second projections, as usual in the classical λ-calculus: $\mathbf{true} \equiv \lambda \mathbf{x}\,\lambda \mathbf{y}\,\mathbf{x}$, $\mathbf{false} \equiv \lambda \mathbf{x}\,\lambda \mathbf{y}\,\mathbf{y}$. Again, note that these are conceived as linear functions, the fact we erase the second/first argument does not mean that the term should be interpreted as a trace out or a measurement. Here is a standard example on how to use them:

$$\mathbf{Not} \equiv \lambda \mathbf{y}\,\Big(\mathbf{y}\,\mathbf{false}\,\mathbf{true}\Big).$$

Notice that this terms globally express a unitary operator, even if some subterms express non unitary ones.

Encoding unary quantum gates. For the Phase gate the naive encoding will not work, i.e.

$$\mathbf{Phase} \not\equiv \lambda \mathbf{y} \left(\mathbf{y}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\mathbf{false}\right)$$

since by bilinearity this would give **Phase false** $\longrightarrow^*$ $e^{i\frac{\pi}{4}}$.**false**, whereas the Phase gate is supposed to place an $e^{i\frac{\pi}{4}}$ only on **true**. The trick is to use abstraction in order to retain the $e^{i\frac{\pi}{4}}$ phase on **true** only (where $\_$ is any base vector, for instance **false**).

$$\mathbf{Phase} \equiv \lambda \mathbf{y} \left(\left(\mathbf{y}\,\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\lambda\mathbf{x}\,\mathbf{false}\right)\,\_\right)$$

Now **Phase true** yields

$$\begin{array}{l}
\lambda \mathbf{y} \left(\left(\mathbf{y}\,\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\lambda\mathbf{x}\,\mathbf{false}\right)\,_\right)\,\mathbf{true} \\
\left(\mathbf{true}\,\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\lambda\mathbf{x}\,\mathbf{false}\right)\,_ \\
\left((\lambda\mathbf{x}\,\lambda\mathbf{y}\,\mathbf{x})\,\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\lambda\mathbf{x}\,\mathbf{false}\right)\,_ \\
(\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true}))\,_ \\
e^{i\frac{\pi}{4}}.\mathbf{true}
\end{array}$$

whereas **Phase false** yields

$$\begin{array}{l}
\lambda \mathbf{y} \left(\left(\mathbf{y}\,\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\lambda\mathbf{x}\,\mathbf{false}\right)\,_\right)\,\mathbf{false} \\
\left(\mathbf{false}\,\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\lambda\mathbf{x}\,\mathbf{false}\right)\,_ \\
\left((\lambda\mathbf{x}\,\lambda\mathbf{y}\,\mathbf{y})\,\lambda\mathbf{x}\,(e^{i\frac{\pi}{4}}.\mathbf{true})\,\lambda\mathbf{x}\,\mathbf{false}\right)\,_ \\
(\lambda\mathbf{x}\,\mathbf{false})\,_ \\
\mathbf{false}
\end{array}$$

This idea of using a dummy abstraction to restrict linearity can be generalized with the following construct: $[\mathbf{t}] \equiv \lambda\mathbf{x}\,\mathbf{t}$, whose effect is to associate a base vector $[\mathbf{t}]$ to any state, and its converse: $\{\mathbf{t}\} \equiv \mathbf{t}\,\_$ where $\_$ is any base vector, for instance **false**. We then have the derived rule $\{[\mathbf{t}]\} \longrightarrow \mathbf{t}$, thus $\{.\}$ is a "left-inverse" of $[.]$, but not a "right inverse", just like eval and $'$ (quote) in LISP. Note that these hooks do not add anymore power to the calculus, in particular they do not enable cloning. We cannot clone a given state $\alpha.\mathbf{t}+\beta.\mathbf{u}$, but we can copy its classical description $[\alpha.\mathbf{t}+\beta.\mathbf{u}]$. For instance the function $\lambda\mathbf{x}\,[\mathbf{x}]$ will never "canonize" anything else than a base vector, because of restriction $(****)$. The phase gate can then be written

$$\mathbf{Phase} \equiv \lambda \mathbf{y}\,\left\{(\mathbf{y}\,[e^{i\frac{\pi}{4}}.\mathbf{true}])\,[\mathbf{false}]\right\}$$

For the Hadamard gate the game is just the same:

$$\mathbf{H} \equiv \lambda \mathbf{y}\,\left\{\mathbf{y}\,[\frac{\sqrt{2}}{2}.(\mathbf{false}+\mathbf{true})]\,[\frac{\sqrt{2}}{2}.(\mathbf{false}-\mathbf{true})]\right\}$$

Encoding tensors. In quantum mechanics, vectors are put together via the bilinear symbol $\otimes$. But because in our calculus application is bilinear, the usual encoding of tuples does just what is needed.

$$\otimes \equiv \lambda\mathbf{x}\,\lambda\mathbf{y}\,\lambda\mathbf{f}\,(\mathbf{f}\,\mathbf{x}\,\mathbf{y}), \quad \pi_1 \equiv \lambda\mathbf{x}\,\lambda\mathbf{y}\,\mathbf{x}, \quad \pi_2 \equiv \lambda\mathbf{x}\,\lambda\mathbf{y}\,\mathbf{y},$$

$$\bigotimes \equiv \lambda\mathbf{f}\,\lambda\mathbf{g}\,\lambda\mathbf{x}\left(\otimes\,\left(\mathbf{f}\,(\pi_1\,\mathbf{x})\right)\,\left(\mathbf{g}\,(\pi_2\,\mathbf{x})\right)\right)$$

E.g. $\mathbf{H}^{\otimes \mathbf{2}} \equiv \left(\bigotimes \mathbf{H}\,\mathbf{H}\right)$. From there on the infix notation for tensors will be used, i.e. $\mathbf{t}\otimes\mathbf{u} \equiv \otimes\,\mathbf{t}\,\mathbf{u}$, $\quad \mathbf{t} \bigotimes \mathbf{u} \equiv \bigotimes\,\mathbf{t}\,\mathbf{u}$.

Encoding the Cnot gate. This binary gate is essentially a classical gate, its encoding is standard.

$$\mathbf{Cnot} \equiv \lambda\mathbf{x}\left((\pi_1\,\mathbf{x}) \otimes \left(\left((\pi_1\,\mathbf{x})\,\left(\mathbf{Not}\,(\pi_2\,\mathbf{x})\right)\right)\,(\pi_2\,\mathbf{x})\right)\right)$$

Expressing the Deutsch-Josza algorithm parametrically. We can now express algorithms parametrically. Here is the well-known simple example of the Deutsch algorithm.

$$\mathbf{Dj}_1 \equiv \lambda\mathbf{x}\left(\mathbf{H}^{\otimes \mathbf{2}}\left(\mathbf{x}\left(\mathbf{H}^{\otimes \mathbf{2}}\,(\mathbf{false}\otimes\mathbf{true})\right)\right)\right)$$

But we can also express control structure and use them to express the dependence of the Deutsch-Josza algorithm with respect to the size of the input. Encoding the natural number n as the Church numeral $\mathbf{n} \equiv \lambda\mathbf{x}\,\lambda\mathbf{f}\,(\mathbf{f}^n\;\mathbf{x})$ the term $(\mathbf{n}\;\mathbf{H}\;\lambda\mathbf{y}\,(\mathbf{H}\bigotimes\mathbf{y}))$ reduces to $\mathbf{H}^{\otimes^{n+1}}$ and similarly the term $(\mathbf{n}\;\mathbf{true}\;\lambda\mathbf{y}\,(\mathbf{false}\otimes\mathbf{y}))$ reduces to $\mathbf{false}^{\otimes^n}\otimes\mathbf{true}$. Thus the expression of the Deutsch-Josza algorithm term of the introduction is now straightforward, see Figure 1.

Infinite dimensional operators. Notice that our language enables us to express operators independently of the dimension of the space they apply to, and even when this dimension is infinite. For instance the identity operator is not expressed as a sum of projections whose number would depend on the dimension of the space, but as the mere lambda term $\lambda\mathbf{x}\,\mathbf{x}$. In this sense our language is a language of infinite dimensional computable linear operators, in the same way that matrices are a language of computable finite dimensional linear operators.

V. CONFLUENCE

The main theorem of this paper is the confluence of the system L. This section is quite technical. A reader who is not familiar with rewriting techniques may be contempt with reading just Definition 6 and Theorem 1, and then skipping to Section VI. A reader with an interest in such techniques may on the other hand find the architecture of the proof quite useful. We shall proceed in two steps and prove first the confluence of the system $R = S \cup E \cup F \cup A$, i.e. the system L minus the rule B. To prove the confluence of R we prove its termination and local

$$\mathbf{Dj} \equiv \lambda\mathbf{n}\lambda\mathbf{x}\left((\mathbf{n}\ \mathbf{H}\ \lambda\mathbf{y}\,(\mathbf{H}\bigotimes\mathbf{y}))\left(\mathbf{x}\left((\mathbf{n}\ \mathbf{H}\ \lambda\mathbf{y}\,(\mathbf{H}\bigotimes\mathbf{y}))\,(\mathbf{n}\ \mathbf{true}\ \lambda\mathbf{y}\,(\mathbf{false}\otimes\mathbf{y}))\right)\right)\right)$$

FIG. 1: Parametric Deutsch-Josza algorithm

confluence. To be able to use a critical pair lemma, we shall use a well-known technique, detailed in the Section V A, and introduce an extension $R_{ext} = S \cup E \cup F_{ext} \cup A$ of the system R as well as a more restricted form of AC-rewriting. This proof will proceed step by step as we shall prove first the local confluence of the system $S \cup E$ (Section V B) then that of $S \cup E \cup F_{ext}$ (Section V C) and finally that of $S \cup E \cup F_{ext} \cup A$ (Section V D). The last step towards our main goal is to show that the $B^{\|}$ rule is strongly confluent on the term algebra, and commutes with R^*, hence giving the confluence of L (Section V E).

To prove the local confluence of the system $S \cup E$ we shall prove that of the system $S_0 \cup E$ where S_0 is a small avatar of S. Then we use a novel proof technique in order to extend from S_0 to S, hereby obtaining the confluence of $S \cup E$. As the system R does not deal at all with lambda abstractions and bound variables, we have, throughout this first part of the proof, considered $\lambda\mathbf{x}$ as a unary function symbol and the bound occurrences of $\mathbf{x}$ as constants. This way we can safely apply known theorems about first-order rewriting.

A. Extensions and the critical pairs lemma

The term $((\mathbf{a}+\mathbf{b})+\mathbf{a})+\mathbf{c}$ is AC-equivalent to $((\mathbf{a}+\mathbf{a})+\mathbf{b})+\mathbf{c}$ and thus reduces to $((1+1).\mathbf{a}+\mathbf{b})+\mathbf{c}$. However, no subterm of $((\mathbf{a}+\mathbf{b})+\mathbf{a})+\mathbf{c}$ matches $\mathbf{u}+\mathbf{u}$. Thus we cannot restrict the application of a rewrite rule to a subterm of the term to be reduced, and we have to consider all the AC-equivalents of this term first. This problem has been solved by [18, 21] that consider a simpler form of application (denoted $\longrightarrow_{X,AC}$) and an extra rule $(\mathbf{u}+\mathbf{u})+\mathbf{x} \longrightarrow (1+1).\mathbf{u}+\mathbf{x}$. Notice that now the term $((\mathbf{a}+\mathbf{b})+\mathbf{a})+\mathbf{c}$ has a subterm $(\mathbf{a}+\mathbf{b})+\mathbf{a}$ that is AC-equivalent to an instance of the left-hand-side of the new rewrite rule.

Definition 4 (The relation $\longrightarrow_{X,AC}$) *Let X be a rewrite system, we define the relation $\longrightarrow_{X,AC}$ as follows $t \longrightarrow_{X,AC} u$ if there exists an occurrence p in t, a rewrite rule $l \longrightarrow r$ in X and a substitution σ such that $t_{|p} =_{AC} \sigma l$ and $u =_{AC} t[p \leftarrow \sigma r]$.*

Definition 5 (The extension rules) $(\alpha.\mathbf{u} + \beta.\mathbf{u}) + \mathbf{x} \longrightarrow (\alpha+\beta).\mathbf{u}+\mathbf{x}$, $\quad(\alpha.\mathbf{u}+\mathbf{u})+\mathbf{x} \longrightarrow (\alpha+1).\mathbf{u}+\mathbf{x}$, $\quad(\mathbf{u}+\mathbf{u})+\mathbf{x} \longrightarrow (1+1).\mathbf{u}+\mathbf{x}$ $\quad(*)$ *We call F_{ext} the system formed by the rules of F and these three rules and R_{ext} the system $S \cup E \cup F_{ext} \cup A$.*

As we shall see the confluence of $\longrightarrow_{R\ AC}$ is a consequence of that of $\longrightarrow_{(R_{ext}),AC}$.

As usual we write $\mathbf{t} \longrightarrow^* \mathbf{u}$ if and only if $\mathbf{t} = \mathbf{u}$ or $\mathbf{t} \longrightarrow \ldots \longrightarrow \mathbf{u}$. We also write $\mathbf{t} \longrightarrow^? \mathbf{u}$ if and only if $\mathbf{t} = \mathbf{u}$ or $\mathbf{t} \longrightarrow \mathbf{u}$.

Definition 6 ((local) confluence) *A relation X is said to be confluent if whenever $t \longrightarrow_X^* u$ and $t \longrightarrow_X^* v$, there exists a term w such that $u \longrightarrow_X^* w$ and $v \longrightarrow_X^* w$. A relation X is said to be locally confluent if whenever $t \longrightarrow_X u$ and $t \longrightarrow_X v$, there exists a term w such that $u \longrightarrow_X^* w$ and $v \longrightarrow_X^* w$.*

Definition 7 (Critical pair) *Let $l \longrightarrow r$ and $l' \longrightarrow r'$ two rewrite rules of an AC-rewrite system X, let p be an occurrence in l such that $l_{|p}$ is not a free variable. Let σ be a AC-unifier for $l_{|p}$ and l', the pair $\big(\sigma r, \sigma(l[p \leftarrow r'])\big)$ is a* critical pair *of the the rewrite system X.*

Proposition 3 (Critical pair lemma) *The relation $\longrightarrow_{X,AC}$ is locally confluent if for each critical pair (t,u) there exists a term w such that $t \longrightarrow_{X,AC}^* w$ $u \longrightarrow_{X,AC}^* w$.*

Proposition 4 *If $\longrightarrow_{R_{ext},AC}$ is locally confluent and $\longrightarrow_{R\ AC}$ terminates then $\longrightarrow_{R\ AC}$ is confluent.*

Proof. From the general Theorems 8.9, 9.3 and 10.5 of [21]. Thus to prove the confluence of $\longrightarrow_{R\ AC}$ we shall prove its termination and the local confluence of $\longrightarrow_{R_{ext},AC}$.

B. Local confluence of $S \cup E$

Definition 8 (The rewrite system S_0) *The system S_0 is formed by the rules*
$0+\alpha \longrightarrow \alpha$, $\quad 0\times\alpha \longrightarrow 0$, $\quad 1\times\alpha \longrightarrow \alpha$, $\quad \alpha\times(\beta+\gamma) \longrightarrow (\alpha\times\beta)+(\alpha\times\gamma)$
where $+$ and $\times$ are AC symbols.

Proposition 5 *The system $S_0 \cup E$ is locally confluent.*

Proof. We check that all the critical pair close using Proposition 3. This can be automatically done using, for instance, the system CIME [9].

Definition 9 (Subsumption) *A terminating and confluent relation S* subsumes *a relation S_0 if whenever $t \longrightarrow_{S_0} u$, t and u have the same S-normal form.*

Definition 10 (Commuting relations) *Two relations X and Y are said to be commuting if whenever $t \longrightarrow_X u$ and $t \longrightarrow_Y v$, there exists a term w such that $u \longrightarrow_Y w$ and $v \longrightarrow_X w$.*

Proposition 6 (The avatar lemma) *[4] Let E, S and S_0 be three relations defined on a set such that:*
-S is terminating and confluent;
-S subsumes S_0;
-$S_0 \cup E$ is locally confluent;
-E commutes with S^*.
Then, the relation $S \cup E$ is locally confluent.

Proof. [*E can be simulated by $E; S\downarrow$*].
If $t \longrightarrow_E u$ and $t \longrightarrow_{S\downarrow} v$, then there exists w such that $u \longrightarrow_{S\downarrow} w$ and $v \longrightarrow_{E;S\downarrow} w$. Indeed by commutation of E and S^* there exists a such that $u \longrightarrow_{S^*} a$ and $v \longrightarrow_E a$. Normalizing a under S yields the w.

[*$S_0 \cup E$ can be simulated by $(E; S\downarrow)^?$*].
If $t \longrightarrow_{S_0 \cup E} u$ and $t \longrightarrow_{S\downarrow} v$, then there exists w such that $u \longrightarrow_{S\downarrow} w$ and $v \longrightarrow^?_{E;S\downarrow} w$. Indeed if $t \longrightarrow_{S_0} u$ this is just subsumption, else the first point of this proof applies.

[*$S \cup E$ can be simulated by $(E; S\downarrow)^?$*].
If $t \longrightarrow_{S \cup E} u$ and $t \longrightarrow_{S\downarrow} v$, then there exists w such that $u \longrightarrow_{S\downarrow} w$ and $v \longrightarrow^?_{E;S\downarrow} w$. Indeed if $t \longrightarrow_S u$ this is just the normalization of S, else the first point of this proof applies.

[*$E; S\downarrow$ is locally confluent*].
If $t \longrightarrow_{E;S\downarrow} u$ and $t \longrightarrow_{E;S\downarrow} v$, then there exists w such that $u \longrightarrow^*_{E;S\downarrow} w$ and $v \longrightarrow^*_{E;S\downarrow} w$. Indeed if $t \longrightarrow_E a \longrightarrow_{S\downarrow} u$ and $t \longrightarrow_E b \longrightarrow_{S\downarrow} v$ we know from the local confluence of $S_0 \cup E$ that there exists c such that $a \longrightarrow^*_{S_0 \cup E} c$ and $b \longrightarrow^*_{S_0 \cup E} c$. Normalizing c under S yields the w. This is because by the second point of the proof $u \longrightarrow^*_{E;S\downarrow} w$ and $v \longrightarrow^*_{E;S\downarrow} w$.

[*$S \cup E$ is locally confluent*].
If $t \longrightarrow_{S \cup E} u$ and $t \longrightarrow_{S \cup E} v$, then there exists w such that $u \longrightarrow^*_{S \cup E} w$ and $v \longrightarrow^*_{S \cup E} w$. Indeed call $t^\downarrow$, $u^\downarrow$, $v^\downarrow$ the S normalized version of t, u, v. By the third point of our proof we have $t^\downarrow \longrightarrow^?_{E;S\downarrow} u^\downarrow$ and $t^\downarrow \longrightarrow^?_{E;S\downarrow} v^\downarrow$. By the fourth point of our proof there exists w such that $u^\downarrow \longrightarrow^*_{E;S\downarrow} w$ and $v^\downarrow \longrightarrow^*_{E;S\downarrow} w$.

Proposition 7 *For any scalar rewrite system S the system $S \cup E$ is locally confluent.*

Proof. The system S is confluent and terminating because it is a scalar rewrite system. The system S subsumes S_0 because S is a scalar rewrite system. From Proposition 5, the system $S_0 \cup E$ is locally confluent. Finally, we check that the system E commutes with S^*. Indeed, we check this for each rule of E, using the fact that in the left member of a rule, each subterm of sort scalar is either a variables or 0 or 1, which are normal forms. We conclude with Proposition 6.

C. Local confluence of $S \cup E \cup F_{ext}$

Proposition 8 *The system $S \cup E \cup F_{ext}$ is locally confluent.*

Proof. This system is made of two subsystems : $S \cup E$ and F_{ext}. To prove that it is locally confluent, we prove that all critical pairs close. We used an AC-unification algorithm to compute these critical pairs. If both rules used are rules of the system $S \cup E$, then the critical pair closes by Proposition 7.

Because of the conditionality of the rewrite system we have to check the 43 other critical pairs by hand.

Pair 1 $\mathbf{0} + \mathbf{0} \longrightarrow (1+1).\mathbf{0}$
$\mathbf{0} + \mathbf{0} \longrightarrow \mathbf{0}$

This critical pair closes on $\mathbf{0}$.

Pair 2 $\alpha.(\beta.\mathbf{u} + \beta'.\mathbf{u}) \longrightarrow \alpha.(\beta.\mathbf{u}) + \alpha.(\beta'.\mathbf{u})$
$\alpha.(\beta.\mathbf{u} + \beta'.\mathbf{u}) \longrightarrow \alpha.((\beta + \beta').\mathbf{u})$

The term $\mathbf{u}$ is closed and normal thus the top reduct further reduces to $(\alpha \times \beta + \alpha \times \beta').\mathbf{u}$ and the bottom reduct further reduces to $(\alpha \times (\beta + \beta').\mathbf{u}$. Since S is a scalar rewrite system these two terms have a common reduct.

Pair 3

$\alpha.(\beta.\mathbf{u} + \beta'.\mathbf{u} + \mathbf{x}) \longrightarrow \alpha.\beta.\mathbf{u} + \alpha.(\beta'.\mathbf{u} + \mathbf{x})$
$\alpha.(\beta.\mathbf{u} + \beta'.\mathbf{u} + \mathbf{x}) \longrightarrow \alpha.(((\beta + \beta').\mathbf{u}) + \mathbf{x})$

The top reduct further reduces to $\alpha.(\beta.\mathbf{u}) + \alpha.(\beta'.\mathbf{u}) + \alpha.\mathbf{x}$ and the bottom reduct further reduces to $\alpha.((\beta + \beta').\mathbf{u}) + \alpha.\mathbf{x}$ hence the situation is analogous to that of Pair 2.

Pair 4 $\alpha.(\beta.\mathbf{u}+\beta'.\mathbf{u}+\mathbf{x}+\mathbf{y}) \longrightarrow \alpha.(\beta.\mathbf{u}+\mathbf{x})+\alpha.(\beta'.\mathbf{u}+\mathbf{y})$
$\alpha.(\beta.\mathbf{u}+\beta'.\mathbf{u}+\mathbf{x}+\mathbf{y}) \longrightarrow \alpha.(((\beta+\beta').\mathbf{u})+\mathbf{x}+\mathbf{y})$

Analogous to Pair 2.

Pair 5 $\alpha.(\beta.\mathbf{u} + \mathbf{u}) \longrightarrow \alpha.(\beta.\mathbf{u}) + \alpha.\mathbf{u}$
$\alpha.(\beta.\mathbf{u} + \mathbf{u}) \longrightarrow \alpha.((\beta + 1).\mathbf{u})$

Analogous to Pair 2.

Pair 6 $\alpha.(\beta.\mathbf{u} + \mathbf{u} + \mathbf{x}) \longrightarrow \alpha.\mathbf{u} + \alpha.(\beta.\mathbf{u} + \mathbf{x})$
$\alpha.(\beta.\mathbf{u} + \mathbf{u} + \mathbf{x}) \longrightarrow \alpha.(((1 + \beta).\mathbf{u}) + \mathbf{x})$

Analogous to Pair 2.

Pair 7 $\alpha.(\beta.\mathbf{u} + \mathbf{u} + \mathbf{x}) \longrightarrow \alpha.(\beta.\mathbf{u}) + \alpha.(\mathbf{u} + \mathbf{x})$
$\alpha.(\beta.\mathbf{u} + \mathbf{u} + \mathbf{x}) \longrightarrow \alpha.(((\beta + 1).\mathbf{u}) + \mathbf{x})$

Analogous to Pair 2.

Pair 8 $\alpha.(\mathbf{u} + \beta.\mathbf{u} + \mathbf{x} + \mathbf{y}) \longrightarrow \alpha.(\mathbf{u} + \mathbf{x}) + \alpha.(\beta.\mathbf{u} + \mathbf{y})$
$\alpha.(\mathbf{u} + \beta.\mathbf{u} + \mathbf{x} + \mathbf{y}) \longrightarrow \alpha.(((1 + \beta).\mathbf{u}) + \mathbf{x} + \mathbf{y})$

Analogous to Pair 2.

Pair 9 $\alpha.(\mathbf{u} + \mathbf{u}) \longrightarrow \alpha.\mathbf{u} + \alpha.\mathbf{u}$
$\alpha.(\mathbf{u} + \mathbf{u}) \longrightarrow \alpha.((1 + 1).\mathbf{u})$

Analogous to Pair 2.

Pair 10 $\alpha.(\mathbf{u} + \mathbf{u} + \mathbf{x}) \longrightarrow \alpha.\mathbf{u} + \alpha.(\mathbf{u} + \mathbf{x})$
$\alpha.(\mathbf{u} + \mathbf{u} + \mathbf{x}) \longrightarrow \alpha.(((1 + 1).\mathbf{u}) + \mathbf{x})$

Analogous to Pair 2.

Pair 11 $\alpha.(\mathbf{u}+\mathbf{u}+\mathbf{x}+\mathbf{y}) \longrightarrow \alpha.(\mathbf{u}+\mathbf{x})+\alpha.(\mathbf{u}+\mathbf{y})$
$\alpha.(\mathbf{u}+\mathbf{u}+\mathbf{x}+\mathbf{y}) \longrightarrow \alpha.(((1+1).\mathbf{u})+\mathbf{x}+\mathbf{y})$

Analogous to Pair 2.

Pair 12 $\alpha.\mathbf{u}+\alpha.\mathbf{u} \longrightarrow (\alpha+\alpha).\mathbf{u}$
$\alpha.\mathbf{u}+\alpha.\mathbf{u} \longrightarrow (1+1).(\alpha.\mathbf{u})$

The bottom reduct further reduces to $((1+1)\times\alpha).\mathbf{u}$. Since S is a scalar rewrite system the two terms have a common reduct.

Pair 13 $\alpha.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+\alpha).\mathbf{u}+\mathbf{x}$
$\alpha.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{x} \longrightarrow (1+1).(\alpha.\mathbf{u})+\mathbf{x}$

Analogous to Pair 12.

Pair 14 $\alpha.\mathbf{u}+\beta.\mathbf{u}+\gamma.\mathbf{u} \longrightarrow (\alpha+\beta).\mathbf{u}+\gamma.\mathbf{u}$
$\alpha.\mathbf{u}+\beta.\mathbf{u}+\gamma.\mathbf{u} \longrightarrow (\alpha+\gamma).\mathbf{u}+\beta.\mathbf{u}$

This critical pair closes on $(\alpha+\beta+\gamma).\mathbf{u}$.

Pair 15 $\alpha.\mathbf{u}+\beta.\mathbf{u}+\gamma.\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+\beta).\mathbf{u}+\gamma.\mathbf{u}+\mathbf{x}$
$\alpha.\mathbf{u}+\beta.\mathbf{u}+\gamma.\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+\gamma).\mathbf{u}+\beta.\mathbf{u}+\mathbf{x}$

Analogous to Pair 14.

Pair 16 $\alpha.\mathbf{u}+\beta.\mathbf{u}+\mathbf{u} \longrightarrow (\alpha+\beta).\mathbf{u}+\mathbf{u}$
$\alpha.\mathbf{u}+\beta.\mathbf{u}+\mathbf{u} \longrightarrow (\alpha+1).\mathbf{u}+\beta.\mathbf{u}$

Analogous to Pair 14.

Pair 17 $\alpha.\mathbf{u}+\beta.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+\beta).\mathbf{u}+\mathbf{u}+\mathbf{x}$
$\alpha.\mathbf{u}+\beta.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+1).\mathbf{u}+\beta.\mathbf{u}+\mathbf{x}$

Analogous to Pair 14.

Pair 18 $\alpha.\mathbf{u}+\alpha.\mathbf{u}+\beta.\mathbf{u} \longrightarrow (1+1).(\alpha.\mathbf{u})+\beta.\mathbf{u}$
$\alpha.\mathbf{u}+\alpha.\mathbf{u}+\beta.\mathbf{u} \longrightarrow (\alpha+\beta).\mathbf{u}+\alpha.\mathbf{u}$

The terms $\mathbf{u}$ is closed and normal thus the top reduct further reduces to $(\alpha\times(1+1)+\beta).\mathbf{u}$ and the bottom reduct further reduces to $(\alpha+\alpha+\beta).\mathbf{u}$. Since S is a scalar rewrite system these two terms have a common reduct.

Pair 19 $\alpha.\mathbf{u}+\alpha.\mathbf{u}+\beta.\mathbf{u}+\mathbf{x} \longrightarrow (1+1).(\alpha.\mathbf{u})+\beta.\mathbf{u}+\mathbf{x}$
$\alpha.\mathbf{u}+\alpha.\mathbf{u}+\beta.\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+\beta).\mathbf{u}+\alpha.\mathbf{u}+\mathbf{x}$

Analogous to Pair 18.

Pair 20 $\alpha.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{u} \longrightarrow (1+1).(\alpha.\mathbf{u})+\mathbf{u}$
$\alpha.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{u} \longrightarrow (\alpha+1).\mathbf{u}+\alpha.\mathbf{u}$

Analogous to Pair 18.

Pair 21 $\alpha.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (1+1).(\alpha.\mathbf{u})+\mathbf{u}+\mathbf{x}$
$\alpha.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+1).\mathbf{u}+\alpha.\mathbf{u}+\mathbf{x}$

Analogous to Pair 18.

Pair 22 $\alpha.\mathbf{u}+\mathbf{u}+\mathbf{u} \longrightarrow (\alpha+1).\mathbf{u}+\mathbf{u}$
$\alpha.\mathbf{u}+\mathbf{u}+\mathbf{u} \longrightarrow (1+1).\mathbf{u}+\alpha.\mathbf{u}$

Analogous to Pair 14.

Pair 23 $\alpha.\mathbf{u}+\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+1).\mathbf{u}+\mathbf{u}+\mathbf{x}$
$\alpha.\mathbf{u}+\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (1+1).\mathbf{u}+\alpha.\mathbf{u}+\mathbf{x}$

Analogous to Pair 14.

Pair 24 $\mathbf{0}.\mathbf{u}+\alpha.\mathbf{u} \longrightarrow \mathbf{0}+\alpha.\mathbf{u}$
$\mathbf{0}.\mathbf{u}+\alpha.\mathbf{u} \longrightarrow (\mathbf{0}+\alpha).\mathbf{u}$

This critical pair closes on $\alpha.\mathbf{u}$.

Pair 25 $\mathbf{0}.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{x} \longrightarrow \mathbf{0}+\alpha.\mathbf{u}+\mathbf{x}$
$\mathbf{0}.\mathbf{u}+\alpha.\mathbf{u}+\mathbf{x} \longrightarrow (\mathbf{0}+\alpha).\mathbf{u}+\mathbf{x}$

Analogous to Pair 24.

Pair 26 $\mathbf{0}.\mathbf{u}+\mathbf{u} \longrightarrow \mathbf{0}+\mathbf{u}$
$\mathbf{0}.\mathbf{u}+\mathbf{u} \longrightarrow (\mathbf{0}+1).\mathbf{u}$

Analogous to Pair 24.

Pair 27 $\mathbf{0}.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow \mathbf{0}+\mathbf{u}+\mathbf{x}$
$\mathbf{0}.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (\mathbf{0}+1).\mathbf{u}+\mathbf{x}$

Analogous to Pair 24.

Pair 28 $\alpha.\mathbf{u}+1.\mathbf{u} \longrightarrow \alpha.\mathbf{u}+\mathbf{u}$
$\alpha.\mathbf{u}+1.\mathbf{u} \longrightarrow (\alpha+1).\mathbf{u}$

The terms $\mathbf{u}$ is closed and normal thus the top reduct further reduces to the bottom reduct.

Pair 29 $\alpha.\mathbf{u}+1.\mathbf{u}+\mathbf{x} \longrightarrow \alpha.\mathbf{u}+\mathbf{u}+\mathbf{x}$
$\alpha.\mathbf{u}+1.\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+1).\mathbf{u}+\mathbf{x}$

Analogous to Pair 28.

Pair 30 $1.\mathbf{u}+\mathbf{u} \longrightarrow \mathbf{u}+\mathbf{u}$
$1.\mathbf{u}+\mathbf{u} \longrightarrow (1+1).\mathbf{u}$

Analogous to Pair 28.

Pair 31 $1.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow \mathbf{u}+\mathbf{u}+\mathbf{x}$
$1.\mathbf{u}+\mathbf{u}+\mathbf{x} \longrightarrow (1+1).\mathbf{u}+\mathbf{x}$

Analogous to Pair 28.

Pair 32 $\mathbf{0}+\alpha.\mathbf{0} \longrightarrow \mathbf{0}+\mathbf{0}$
$\mathbf{0}+\alpha.\mathbf{0} \longrightarrow (1+\alpha).\mathbf{0}$

This critical pair closes on $\mathbf{0}$.

Pair 33 $\mathbf{0}+\alpha.\mathbf{0}+\mathbf{x} \longrightarrow \mathbf{0}+\mathbf{0}+\mathbf{x}$
$\mathbf{0}+\alpha.\mathbf{0}+\mathbf{x} \longrightarrow (1+\alpha).\mathbf{0}+\mathbf{x}$

Analogous to Pair 32.

Pair 34 $\alpha.\mathbf{0}+\beta.\mathbf{0} \longrightarrow \mathbf{0}+\beta.\mathbf{0}$
$\alpha.\mathbf{0}+\beta.\mathbf{0} \longrightarrow (\alpha+\beta).\mathbf{0}$

Analogous to Pair 28.

Pair 35 $\alpha.\mathbf{0}+\beta.\mathbf{0}+\mathbf{x} \longrightarrow \mathbf{0}+\beta.\mathbf{0}+\mathbf{x}$
$\alpha.\mathbf{0}+\beta.\mathbf{0}+\mathbf{x} \longrightarrow (\alpha+\beta).\mathbf{0}+\mathbf{x}$

Analogous to Pair 28.

Pair 36 $\alpha.\gamma.\mathbf{u}+\beta.\gamma.\mathbf{u} \longrightarrow (\alpha\times\gamma).\mathbf{u}+\beta.\gamma.\mathbf{u}$
$\alpha.\gamma.\mathbf{u}+\beta.\gamma.\mathbf{u} \longrightarrow (\alpha+\beta).\gamma.\mathbf{u}$

The terms $\mathbf{u}$ is closed and normal thus the top reduct reduces to $(\alpha\times\gamma+\beta\times\gamma).\mathbf{u}$ and the bottom reduct further reduces to $((\alpha+\beta)\times\gamma).\mathbf{u}$. Since S is a scalar rewrite system the two terms have a common reduct.

Pair 37 $\alpha.\gamma.\mathbf{u}+\beta.\gamma.\mathbf{u}+\mathbf{x} \longrightarrow (\alpha\times\gamma).\mathbf{u}+\beta.\gamma.\mathbf{u}+\mathbf{x}$
$\alpha.\gamma.\mathbf{u}+\beta.\gamma.\mathbf{u}+\mathbf{x} \longrightarrow (\alpha+\beta).\gamma.\mathbf{u}+\mathbf{x}$

Analogous to Pair 36.

Pair 38 $\beta.\mathbf{u}+\alpha.\beta.\mathbf{u} \longrightarrow \beta.\mathbf{u}+(\alpha\times\beta).\mathbf{u}$
$\beta.\mathbf{u}+\alpha.\beta.\mathbf{u} \longrightarrow (1+\alpha).\beta.\mathbf{u}$

Analogous to Pair 36.

Pair 39 $\beta.\mathbf{u}+\alpha.\beta.\mathbf{u}+\mathbf{x} \longrightarrow \beta.\mathbf{u}+(\alpha\times\beta).\mathbf{u}+\mathbf{x}$
$\beta.\mathbf{u}+\alpha.\beta.\mathbf{u}+\mathbf{x} \longrightarrow (1+\alpha).\beta.\mathbf{u}+\mathbf{x}$

Analogous to Pair 36.

Pair 40 $\alpha.(\mathbf{u}+\mathbf{v})+\beta.(\mathbf{u}+\mathbf{v}) \longrightarrow \alpha.\mathbf{u}+\alpha.\mathbf{v}+\beta.(\mathbf{u}+\mathbf{v})$
$\alpha.(\mathbf{u}+\mathbf{v})+\beta.(\mathbf{u}+\mathbf{v}) \longrightarrow (\alpha+\beta).(\mathbf{u}+\mathbf{v})$

The terms $\mathbf{u}+\mathbf{v}$ is closed and normal and hence so are $\mathbf{u}$ and $\mathbf{v}$. Thus the top reduct further reduces to $\alpha.\mathbf{u}+\alpha.\mathbf{v}+\beta.\mathbf{u}+\beta.\mathbf{v}$ and then $(\alpha+\beta).\mathbf{u}+(\alpha+\beta).\mathbf{v}$ the bottom reduct further reduces to this same term.

Pair 41 $\alpha.(\mathbf{u}+\mathbf{v})+\beta.(\mathbf{u}+\mathbf{v})+\mathbf{x} \longrightarrow \alpha.\mathbf{u}+\alpha.\mathbf{v}+\beta.(\mathbf{u}+\mathbf{v})+\mathbf{x}$
$\alpha.(\mathbf{u}+\mathbf{v})+\beta.(\mathbf{u}+\mathbf{v})+\mathbf{x} \longrightarrow (\alpha+\beta).(\mathbf{u}+\mathbf{v})+\mathbf{x}$

Analogous to Pair 40.

Pair 42 $(\mathbf{u}+\mathbf{v})+\alpha.(\mathbf{u}+\mathbf{v}) \longrightarrow (\mathbf{u}+\mathbf{v})+\alpha.\mathbf{u}+\alpha.\mathbf{v}$
$(\mathbf{u}+\mathbf{v})+\alpha.(\mathbf{u}+\mathbf{v}) \longrightarrow (\alpha+1).(\mathbf{u}+\mathbf{v})$

Analogous to Pair 40.

Pair 43 $(\mathbf{u}+\mathbf{v})+\alpha.(\mathbf{u}+\mathbf{v})+\mathbf{x} \longrightarrow (\mathbf{u}+\mathbf{v})+\alpha.\mathbf{u}+\alpha.\mathbf{v}+\mathbf{x}$
$(\mathbf{u}+\mathbf{v})+\alpha.(\mathbf{u}+\mathbf{v})+\mathbf{x} \longrightarrow (\alpha+1).(\mathbf{u}+\mathbf{v})+\mathbf{x}$

Analogous to Pair 40.

D. Local confluence and confluence of R

Proposition 9 *The system $R = S\cup E\cup F_{ext}\cup A$ is locally confluent.*

Proof. This system is made of two subsystems: $S\cup E\cup F_{ext}$ and A. To prove that it is locally confluent, we prove that all critical pairs close. If both rules used are rules of the system $S\cup E\cup F_{ext}$, then the critical pair closes by Proposition 8. It is not possible that the top-level rule is in $S\cup E\cup F_{ext}$ and the other in A since the rules of $S\cup E\cup F_{ext}$ do not contain any application. Thus the top-level rule must be in A and the $(S\cup E\cup F_{ext})$-reduction must be performed in a non-toplevel non-variable subterm of the left-hand-side of a rule of A. By inspection of the left-hand-sides of rules $S\cup E\cup F_{ext}$ the subterm must be of the form $\mathbf{u}+\mathbf{v}$, $\alpha.\mathbf{u}$ or $\mathbf{0}$. But this subterm cannot be of the form $\mathbf{u}+\mathbf{v}$, because, by restriction (**), the term itself would not be A-reducible. It cannot be $\mathbf{0}$ since this is normal. Thus it is of the form $\alpha.\mathbf{u}$. As there are five rules reducing a term of this form, there are 10 critical pairs to check. Because of the conditionality of the rewrite system we have to check them by hand.

Pair 1 $(0.\mathbf{u})\,\mathbf{v} \longrightarrow \mathbf{0}\,\mathbf{v}$
$(0.\mathbf{u})\,\mathbf{v} \longrightarrow 0.(\mathbf{u}\,\mathbf{v})$
This critical pair closes on $\mathbf{0}$.

Pair 2 $(1.\mathbf{u})\,\mathbf{v} \longrightarrow \mathbf{u}\,\mathbf{v}$ $(1.\mathbf{u})\,\mathbf{v} \longrightarrow 1.(\mathbf{u}\,\mathbf{v})$
This critical pair closes on $\mathbf{u}\,\mathbf{v}$.

Pair 3 $(\alpha.\mathbf{0})\,\mathbf{v} \longrightarrow \mathbf{0}\,\mathbf{v}$
$(\alpha.\mathbf{0})\,\mathbf{v} \longrightarrow \alpha.(\mathbf{0}\,\mathbf{v})$
This critical pair closes on $\mathbf{0}$.

Pair 4 $(\alpha.(\beta.\mathbf{u}))\,\mathbf{v} \longrightarrow ((\alpha\times\beta).\mathbf{u})\,\mathbf{v}$
$(\alpha.(\beta.\mathbf{u}))\,\mathbf{v} \longrightarrow \alpha.((\beta.\mathbf{u})\,\mathbf{v})$
The term $\mathbf{u}$ is closed and normal by $(***)$. Hence, the critical pair closes on $(\alpha\times\beta).(\mathbf{u}\,\mathbf{v})$.

Pair 5 $(\alpha.(\mathbf{u}+\mathbf{v}))\,\mathbf{w} \longrightarrow (\alpha.\mathbf{u}+\alpha.\mathbf{v})\,\mathbf{w}$
$(\alpha.(\mathbf{u}+\mathbf{v}))\,\mathbf{w} \longrightarrow \alpha.((\mathbf{u}+\mathbf{v})\,\mathbf{w})$
The term $\mathbf{u}+\mathbf{v}$ is closed and normal. Hence, by proposition 2 it is of the form $\sum_i \beta_i.\mathbf{a}_i+\sum_i \mathbf{b}_i$. Therefore the top reduct reduces to $(\sum_i(\alpha\times\beta_i)\downarrow .\mathbf{a}_i+\sum_i \alpha.\mathbf{b}_i)\,\mathbf{w}$, where $\downarrow$ denotes normalization by S. We treat only the case where the terms $(l\times\beta_i)\downarrow$ are neither 0 nor 1, the other cases being similar. Hence, we can apply rules of group A yielding $\sum_i(\alpha\times\beta_i)\downarrow .(\mathbf{a}_i\,\mathbf{w})+\sum_i \alpha.(\mathbf{b}_i\,\mathbf{w})$. It is routine to check that the bottom reduct also reduces to this term.

The five next critical pairs are the symmetrical cases, permuting the left and right-hand-sides of the application.

Now, when both rules are in the group A, there are 9 critical pairs to check.

Pair 11 $(\mathbf{u}+\mathbf{v})(\mathbf{w}+\mathbf{x}) \longrightarrow \mathbf{u}(\mathbf{w}+\mathbf{x})+\mathbf{v}(\mathbf{w}+\mathbf{x})$
$(\mathbf{u}+\mathbf{v})(\mathbf{w}+\mathbf{x}) \longrightarrow (\mathbf{u}+\mathbf{v})\mathbf{w}+(\mathbf{u}+\mathbf{v})\mathbf{x}$
As $\mathbf{u}+\mathbf{v}$ and $\mathbf{w}+\mathbf{x}$ are normal and closed, so are $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ and $\mathbf{x}$. Hence the critical pair closes on $\mathbf{uw}+\mathbf{ux}+\mathbf{vw}+\mathbf{vx}$.

Pair 12 $(\mathbf{u}+\mathbf{v})(\alpha.\mathbf{w}) \longrightarrow \mathbf{u}(\alpha.\mathbf{w})+\mathbf{v}(\alpha.\mathbf{w})$
$(\mathbf{u}+\mathbf{v})(\alpha.\mathbf{w}) \longrightarrow \alpha.((\mathbf{u}+\mathbf{v}).\mathbf{w})$
As, by $(**)$ and $(***)$, $\mathbf{u}+\mathbf{v}$ and $\mathbf{w}$ are closed normal terms , so are $\mathbf{u}$ and $\mathbf{v}$. Thus the top reduct further reduces to $\alpha.(\mathbf{u}\,\mathbf{w})+\alpha.(\mathbf{v}\,\mathbf{w})$ and the bottom reduct further reduces to $\alpha.((\mathbf{u}\,\mathbf{w})+(\mathbf{v}\,\mathbf{w}))$ and both term reduce to $\alpha.(\mathbf{u}\,\mathbf{w})+\alpha.(\mathbf{v}\,\mathbf{w})$.

Pair 13 $(\mathbf{u}+\mathbf{v})\mathbf{0} \longrightarrow \mathbf{0}$
$(\mathbf{u}+\mathbf{v})\mathbf{0} \longrightarrow (\mathbf{u0})+(\mathbf{v0})$
This critical pair closes on $\mathbf{0}$.

Pair 14 $(\alpha.\mathbf{u})(\mathbf{v}+\mathbf{w}) \longrightarrow \alpha.(\mathbf{u}(\mathbf{v}+\mathbf{w}))$
$(\alpha.\mathbf{u})(\mathbf{v}+\mathbf{w}) \longrightarrow (\alpha.\mathbf{u})\mathbf{v}+(\alpha.\mathbf{u})\mathbf{w}$
The terms $\mathbf{u}$ and $\mathbf{v}+\mathbf{w}$ are closed normal. Thus, the top reduct further reduces to $\alpha.(\mathbf{uv}+\mathbf{uw})$ and the bottom reduct to $\alpha.(\mathbf{uv})+\alpha.(\mathbf{uw})$. Hence the critical pair closes on $\alpha.(\mathbf{uv})+\alpha.(\mathbf{uw})$.

Pair 15 $(\alpha.\mathbf{u})(\beta.\mathbf{v}) \longrightarrow \alpha.(\mathbf{u}(\beta.\mathbf{v}))$
$(\alpha.\mathbf{u})(\beta.\mathbf{v}) \longrightarrow \beta.((\alpha.\mathbf{u})\mathbf{v})$
As $\mathbf{u}$ and $\mathbf{v}$ are closed normal, the first term reduces to $\alpha.(\beta.(\mathbf{uv}))$ and the second to $\beta.(\alpha.(\mathbf{uv}))$ and both terms reduce to $(\alpha \times \beta).(\mathbf{uv})$.

Pair 16 $(\alpha.\mathbf{u})\mathbf{0} \longrightarrow \alpha.(\mathbf{u0})$
$(\alpha.\mathbf{u})\mathbf{0} \longrightarrow \mathbf{0}$
This critical pair closes on $\mathbf{0}$.

Pair 17 $\mathbf{0}(\mathbf{u}+\mathbf{v}) \longrightarrow \mathbf{0}$
$\mathbf{0}(\mathbf{u}+\mathbf{v}) \longrightarrow (\mathbf{0u}) + (\mathbf{0v})$
This critical pair closes on $\mathbf{0}$.

Pair 18 $\mathbf{0}(\alpha.\mathbf{u}) \longrightarrow \mathbf{0}$
$\mathbf{0}(\alpha.\mathbf{u}) \longrightarrow \alpha.(\mathbf{0u})$
This critical pair closes on $\mathbf{0}$.

Pair 19 $\mathbf{00} \longrightarrow \mathbf{0}$
$\mathbf{00} \longrightarrow \mathbf{0}$
This critical pair closes on $\mathbf{0}$.

Proposition 10 *The system R terminates.*

Proof. [*The system $E \cup F \cup A$ terminates*]
Consider the following interpretation (compatible with AC)

$$|(\mathbf{u}\ \mathbf{v})| = (3|\mathbf{u}|+2)(3|\mathbf{v}|+2)$$

$$|\mathbf{u}+\mathbf{v}| = 2 + |\mathbf{u}| + |\mathbf{v}|$$

$$|\alpha.\mathbf{u}| = 1 + 2|\mathbf{u}|$$

$$|\mathbf{0}| = 0$$

Each time a term $\mathbf{t}$ rewrites to a term $\mathbf{t}'$ we have $|\mathbf{t}| > |\mathbf{t}'|$. Hence, the system terminates.

[*The system R terminates*]
The system R is $S \cup E \cup F \cup A$. It is formed of two subsystems S and $E \cup F \cup A$. By definition of the function $|\ |$, if a term $\mathbf{t}$ S-reduces to a term $\mathbf{t}'$ then $|\mathbf{t}| = |\mathbf{t}'|$. Consider a R-reduction sequence. At each $E \cup F \cup A$-reduction step, the measure of the term strictly decreases and at each S-reduction step it remains the same. Thus there are only a finite number of $E \cup F \cup A$-reduction steps in the sequence and, as S terminates, the sequence is finite.

Proposition 11 *The system R is confluent.*

Proof. The relation $\longrightarrow_{R_{ext},AC}$ is locally confluent, and $\longrightarrow_{R\ AC}$ terminates, hence $\longrightarrow_{R\ AC}$ is confluent by proposition 4.

E. The system L

We now want to prove that the system L is confluent. With the introduction of the rule B, we lose termination, hence we cannot use Newman's lemma [19] anymore. Thus we shall use for this last part techniques coming from the proof of confluence of the λ-calculus and prove that the relation $\longrightarrow_B^{\parallel}$ is strongly confluent. In our case as we have to mix the rule B with R we shall also prove that it commutes with $\longrightarrow_R^*$.

Definition 11 (Strong confluence) *A relation X is said to be strongly confluent if whenever $t \longrightarrow_X u$ and $t \longrightarrow_X v$, there exists a term w such that $u \longrightarrow_X w$ and $v \longrightarrow_X w$.*

Definition 12 (The relation $\longrightarrow_B^{\parallel}$) *The relation $\longrightarrow_B^{\parallel}$ is the smallest reflexive congruence such that if $\mathbf{t} \longrightarrow_B^{\parallel} \mathbf{t}'$ and $\mathbf{u} \longrightarrow_B^{\parallel} \mathbf{u}'$ then*

$$(\lambda\mathbf{x}\ \mathbf{t})\ \mathbf{u} \longrightarrow_B^{\parallel} \mathbf{t}'[\mathbf{u}'/\mathbf{x}]$$

Note that as a parallelization of the B reduction relation, the $B^{\parallel}$ is also restricted to base vectors.

Proposition 12 *If $\mathbf{v}_1 \longrightarrow_R^* \mathbf{w}_1$ then $\mathbf{v}_1[\mathbf{b}/\mathbf{x}] \longrightarrow_R^* \mathbf{w}_1[\mathbf{b}/\mathbf{x}]$, where $\mathbf{b}$ is a base vector.*

Proof. We need to check is that if the reduction of $\mathbf{v}_1$ to $\mathbf{w}_1$ involves an application of a conditional rule, then the condition is preserved on $\mathbf{v}_1[\mathbf{v}_2/\mathbf{x}]$. Indeed substituting some term in a closed normal term yields the same term.

Proposition 13 *If $\mathbf{v}_2 \longrightarrow_R^* \mathbf{w}_2$ then $\mathbf{v}_1[\mathbf{v}_2/\mathbf{x}] \longrightarrow_R^* \mathbf{v}_1[\mathbf{w}_2/\mathbf{x}]$.*

Proof. The reduction is a congruence.

Proposition 14 *If $\mathbf{t} = \mathbf{u}$ or $\mathbf{t} \longrightarrow_R \mathbf{u}$ and if $\mathbf{t} \longrightarrow_B^{\parallel} \mathbf{v}$ then there exists $\mathbf{w}$ such that $\mathbf{u} \longrightarrow_B^{\parallel} \mathbf{w}$ and $\mathbf{v} \longrightarrow_R^* \mathbf{w}$.*

Proof. By induction on the structure of $\mathbf{t}$. If $\mathbf{t} = \mathbf{u}$ we just take $\mathbf{w} = \mathbf{v}$. Thus we focus in the rest of the proof to the case where $\mathbf{t} \longrightarrow_R \mathbf{u}$.

If the $B^{\parallel}$-reduction takes place at toplevel, then $\mathbf{t} = (\lambda\mathbf{x}\,\mathbf{t}_1)\ \mathbf{t}_2$, $\mathbf{t}_2$ is a base vector and there exists terms $\mathbf{v}_1$ and $\mathbf{v}_2$ such that $\mathbf{t}_1 \longrightarrow_B^{\parallel} \mathbf{v}_1$, $\mathbf{t}_2 \longrightarrow_B^{\parallel} \mathbf{v}_2$ and $\mathbf{v} = \mathbf{v}_1[\mathbf{v}_2/\mathbf{x}]$. Neither $\lambda\mathbf{x}\,\mathbf{t}_1$ nor $\mathbf{t}_2$ is a sum, a product by a scalar or the null vector, hence the R-reduction is just an application of the congruence thus there exists terms $\mathbf{u}_1$ and $\mathbf{u}_2$ such that and $\mathbf{t}_1 \longrightarrow_R^? \mathbf{u}_1$ and $\mathbf{t}_2 \longrightarrow_R^? \mathbf{u}_2$. Since $\mathbf{t}_2$ is a base vector, $\mathbf{u}_2$ is also a base vector. By induction hypothesis, there exist terms $\mathbf{w}_1$ and $\mathbf{w}_2$ such that $\mathbf{u}_1 \longrightarrow_B^{\parallel} \mathbf{w}_1$, $\mathbf{v}_1 \longrightarrow_R^* \mathbf{w}_1$, $\mathbf{u}_2 \longrightarrow_B^{\parallel} \mathbf{w}_2$ and $\mathbf{v}_2 \longrightarrow_R^* \mathbf{w}_2$. We take $\mathbf{w} = \mathbf{w}_1[\mathbf{w}_2/\mathbf{x}]$. We have $(\lambda\mathbf{x}\,\mathbf{u}_1)\ \mathbf{u}_2 \longrightarrow_B^{\parallel} \mathbf{w}$ and by Proposition 12 and 13 we also have $\mathbf{v}_1[\mathbf{v}_2/\mathbf{x}] \longrightarrow_R^* \mathbf{w}$.

If the R-reduction takes place at toplevel, we have to distinguish several cases according to the rule used for this reduction.

- If $\mathbf{t} = \mathbf{t}_1 + \mathbf{0}$ and $\mathbf{u} = \mathbf{t}_1$, then there exists a term $\mathbf{v}_1$ such that $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$ and $\mathbf{v} = \mathbf{v}_1 + \mathbf{0}$. We take $\mathbf{w} = \mathbf{v}_1$.
- If $\mathbf{t} = 0.\mathbf{t}_1$ and $\mathbf{u} = \mathbf{0}$, then there exists a term $\mathbf{v}_1$ such that $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$ and $\mathbf{v} = 0.\mathbf{v}_1$. We take $\mathbf{w} = \mathbf{0}$.
- If $\mathbf{t} = 1.\mathbf{t}_1$ and $\mathbf{u} = \mathbf{t}_1$, then there exists a term $\mathbf{v}_1$ such that $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$ and $\mathbf{v} = 1.\mathbf{v}_1$. We take $\mathbf{w} = \mathbf{v}_1$.
- If $\mathbf{t} = \alpha.\mathbf{0}$ and $\mathbf{u} = \mathbf{0}$, then $\mathbf{v} = \mathbf{t}$. We take $\mathbf{w} = \mathbf{0}$.
- If $\mathbf{t} = \alpha.(\beta.\mathbf{t}_1)$ and $\mathbf{u} = (\alpha\times\beta).\mathbf{t}_1$, then there exists a term $\mathbf{v}_1$ such that $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$ and $\mathbf{v} = \alpha.(\beta.\mathbf{v}_1)$. We take $\mathbf{w} = (\alpha \times \beta).\mathbf{v}_1$.
- If $\mathbf{t} = \alpha.(\mathbf{t}_1 + \mathbf{t}_2)$ and $\mathbf{u} = \alpha.\mathbf{t}_1 + \alpha.\mathbf{t}_2$, then there exist terms $\mathbf{v}_1$ and $\mathbf{v}_2$ such that $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$ and $\mathbf{v} = \alpha.(\mathbf{v}_1 + \mathbf{v}_2)$. We take $\mathbf{w} = \alpha.\mathbf{v}_1 + \alpha.\mathbf{v}_2$.
- If $\mathbf{t} = \alpha.\mathbf{t}_1 + \beta.\mathbf{t}_1$ and $\mathbf{u} = (\alpha + \beta).\mathbf{t}_1$, then by $(*)$ $\mathbf{t}_1$ is L-normal, thus $\mathbf{v} = \mathbf{t}$. We take $\mathbf{w} = \mathbf{u}$.

  The cases of the two other factorisation rules are similar.
- If $\mathbf{t} = (\mathbf{t}_1 + \mathbf{t}_2)\ \mathbf{t}_3$ and $\mathbf{u} = \mathbf{t}_1\ \mathbf{t}_3 + \mathbf{t}_2\ \mathbf{t}_3$, then by $(**)$ the term $\mathbf{t}_1 + \mathbf{t}_2$ is L-normal. There exists a term $\mathbf{v}_3$ such that $\mathbf{t}_3 \longrightarrow_B^{\|} \mathbf{v}_3$ and $\mathbf{v} = (\mathbf{t}_1 + \mathbf{t}_2)\ \mathbf{v}_3$. We take $\mathbf{w} = \mathbf{t}_1\ \mathbf{v}_3 + \mathbf{t}_2\ \mathbf{v}_3$.
- If $\mathbf{t} = (\alpha.\mathbf{t}_1)\ \mathbf{t}_2$ and $\mathbf{u} = \alpha.(\mathbf{t}_1\ \mathbf{t}_2)$, then by $(***)$ $\mathbf{t}_1$ is L-normal. There exists a term $\mathbf{v}_2$ such that $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$ and $\mathbf{v} = (\alpha.\mathbf{t}_1)\ \mathbf{v}_2$. We take $\mathbf{w} = \alpha.(\mathbf{t}_1\ \mathbf{v}_2)$.
- If $\mathbf{t} = \mathbf{0}\ \mathbf{t}_2$ and $\mathbf{u} = \mathbf{0}$, then there exists a term $\mathbf{v}_2$ such that $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$ and $\mathbf{v} = \mathbf{0}\ \mathbf{v}_2$. We take $\mathbf{w} = \mathbf{0}$.

  The three other cases where a rule of group A is applied are symmetric.

Finally if both reductions are just applications of the congruence we apply the induction hypothesis to the subterms.

Proposition 15 ($\longrightarrow_R^*$ commutes with $\longrightarrow_B^{\|}$) *If* $\mathbf{t} \longrightarrow_R^* \mathbf{u}$ *and* $\mathbf{t} \longrightarrow_B^{\|} \mathbf{v}$ *then there exists* $\mathbf{w}$ *such that* $\mathbf{u} \longrightarrow_B^{\|} \mathbf{w}$ *and* $\mathbf{v} \longrightarrow_R^* \mathbf{w}$.

Proof. By induction on the length of the $\longrightarrow_R^*$ derivation. If $\mathbf{t} = \mathbf{u}$ then we take $\mathbf{w} = \mathbf{v}$. Otherwise there exists a term $\mathbf{t}_1$ such that $\mathbf{t} \longrightarrow_R \mathbf{t}_1 \longrightarrow_R^* \mathbf{u}$ with a shorter reduction from $\mathbf{t}_1$ to $\mathbf{u}$. Using proposition 14, there exists a term $\mathbf{w}_1$ such that $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{w}_1$ and $\mathbf{v} \longrightarrow_R^* \mathbf{w}_1$. By induction hypothesis, there exists a term $\mathbf{w}$ such that $\mathbf{u} \longrightarrow_B^{\|} \mathbf{w}$ and $\mathbf{w}_1 \longrightarrow_R^* \mathbf{w}$. We have $\mathbf{u} \longrightarrow_B^{\|} \mathbf{w}$ and $\mathbf{v} \longrightarrow_R^* \mathbf{w}$.

Proposition 16 (Substitution for $B^{\|}$) *If* $\mathbf{t} \longrightarrow_B^{\|} \mathbf{t}'$ *and* $\mathbf{b} \longrightarrow_B^{\|} \mathbf{b}'$ *then* $\mathbf{t}[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{b}'[\mathbf{b}'/\mathbf{x}]$. *Here* $\mathbf{b}$ *denotes a base vector.*

Proof. By induction on the structure of $\mathbf{t}$.

- If $\mathbf{t} = \mathbf{x}$ then $\mathbf{t}' = \mathbf{x}$ and hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = \mathbf{b} \longrightarrow_B^{\|} \mathbf{b}' = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.
- If $\mathbf{t} = \mathbf{y}$ then $\mathbf{t}' = \mathbf{y}$ and hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = \mathbf{y} = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.
- If $\mathbf{t} = \lambda\mathbf{y}\,\mathbf{t}_1$ the $B^{\|}$-reduction is just an application of the congruence. We have $\mathbf{t}' = \lambda\mathbf{y}\,.\mathbf{t}_1'$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{t}_1'$ and the induction hypothesis tells us that $\mathbf{t}_1[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_1'[\mathbf{b}'/\mathbf{x}]$. Hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = \lambda\mathbf{y}\,\mathbf{t}_1[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \lambda\mathbf{y}\,\mathbf{t}_1'[\mathbf{b}'/\mathbf{x}] = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.
- If $\mathbf{t} = (\mathbf{t}_1\ \mathbf{t}_2)$ then we consider two cases.
  - We have $\mathbf{t}_1 = \lambda\mathbf{y}\ \mathbf{t}_3$, $\mathbf{t}_2$ a base state, and $\mathbf{t}' = \mathbf{t}_3'[\mathbf{t}_2'/\mathbf{x}]$, i.e. a B-reduction occurs at top-level. By induction hypothesis we know that $\mathbf{t}_3[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_3'[\mathbf{b}'/\mathbf{x}]$ and $\mathbf{t}_2[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_2'[\mathbf{b}'/\mathbf{x}]$. Because $\mathbf{t}_2$ and $\mathbf{b}$ are base vectors, so is $\mathbf{t}_2[\mathbf{b}/\mathbf{x}]$. Hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = (\lambda\mathbf{y}\,\mathbf{t}_3[\mathbf{b}/\mathbf{x}])\mathbf{t}_2[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_3'[\mathbf{b}'/\mathbf{x}][\mathbf{t}_2'[\mathbf{b}'/\mathbf{x}]/\mathbf{y}] = \mathbf{t}_3'[\mathbf{t}_2'/\mathbf{y}][\mathbf{b}'/\mathbf{x}] = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.
  - If $\mathbf{t}' = (\mathbf{t}_1'\ \mathbf{t}_2')$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{t}_1'$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{t}_2'$, then by induction hypothesis we know that $\mathbf{t}_1[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_1'[\mathbf{b}'/\mathbf{x}]$ and $\mathbf{t}_2[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_2'[\mathbf{b}'/\mathbf{x}]$. Hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = (\mathbf{t}_1[\mathbf{b}/\mathbf{x}]\ \mathbf{t}_2[\mathbf{b}/\mathbf{x}]) \longrightarrow_B^{\|} (\mathbf{t}_1'[\mathbf{b}'/\mathbf{x}]\ \mathbf{t}_2'[\mathbf{b}'/\mathbf{x}]) = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.
- If $\mathbf{t} = \mathbf{0}$ then $\mathbf{t}' = \mathbf{0}$ and hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = \mathbf{0} = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.
- If $\mathbf{t}$ is a sum the $B^{\|}$-reduction is just an application of the congruence. Therefore $\mathbf{t}$ is AC equivalent to $\mathbf{t}_1 + \mathbf{t}_2$ and $\mathbf{t}'$ is AC equivalent to $\mathbf{t}_1' + \mathbf{t}_2'$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{t}_1'$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{t}_2'$. Then by induction hypothesis we know that $\mathbf{t}_1[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_1'[\mathbf{b}'/\mathbf{x}]$ and $\mathbf{t}_2[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_2'[\mathbf{b}'/\mathbf{x}]$. Hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = \mathbf{t}_1[\mathbf{b}/\mathbf{x}] + \mathbf{t}_2[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_1'[\mathbf{b}'/\mathbf{x}] + \mathbf{t}_2'[\mathbf{b}'/\mathbf{x}] = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.
- If $\mathbf{t} = \alpha.\mathbf{t}_1$ the $B^{\|}$-reduction is just an application of the congruence. We have $\mathbf{t}' = \alpha.\mathbf{t}_1'$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{t}_1'$ and the induction hypothesis tells us that $\mathbf{t}_1[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{t}_1'[\mathbf{b}'/\mathbf{x}]$. Hence $\mathbf{t}[\mathbf{b}/\mathbf{x}] = \alpha.\mathbf{t}_1[\mathbf{b}/\mathbf{x}] \longrightarrow_B^{\|} \alpha.\mathbf{t}_1'[\mathbf{b}'/\mathbf{x}] = \mathbf{t}'[\mathbf{b}'/\mathbf{x}]$.

Proposition 17 (Strong confluence of $B^{\|}$) *If* $\mathbf{t} \longrightarrow_B^{\|} \mathbf{u}$ *and* $\mathbf{t} \longrightarrow_B^{\|} \mathbf{v}$ *then there exists* $\mathbf{w}$ *such that* $\mathbf{u} \longrightarrow_B^{\|} \mathbf{w}$ *and* $\mathbf{v} \longrightarrow_B^{\|} \mathbf{w}$.

Proof. By induction on the structure of $\mathbf{t}$.

- If $\mathbf{t}$ is a variable then $\mathbf{u} = \mathbf{t}$ and $\mathbf{v} = \mathbf{t}$. We take $\mathbf{w} = \mathbf{t}$.
- If $\mathbf{t} = \mathbf{0}$ then $\mathbf{u} = \mathbf{0}$ and $\mathbf{v} = \mathbf{0}$. We take $\mathbf{w} = \mathbf{0}$.
- If $\mathbf{t} = \lambda\mathbf{x}\ \mathbf{t}_1$ then $\mathbf{u} = \lambda\mathbf{x}\ \mathbf{u}_1$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{u}_1$ and $\mathbf{v} = \lambda\mathbf{x}\ \mathbf{v}_1$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$. By induction hypothesis, there exists a $\mathbf{w}_1$ such that $\mathbf{u}_1 \longrightarrow_B^{\|} \mathbf{w}_1$ and $\mathbf{v}_1 \longrightarrow_B^{\|} \mathbf{w}_1$. We take $\mathbf{w} = \lambda\mathbf{x}\ \mathbf{w}_1$.
- If $\mathbf{t} = (\mathbf{t}_1\ \mathbf{t}_2)$ then we consider two cases.
  - If the term $\mathbf{t}_1$ has the form $\lambda\mathbf{x}\ \mathbf{t}_3$ and $\mathbf{t}_2$ is a base vector. We consider three subcases, according to the form of the $B^{\|}$-reductions. Either $\mathbf{v} = (\mathbf{v}_1\ \mathbf{v}_2)$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$, and $\mathbf{u} = (\mathbf{u}_1\ \mathbf{u}_2)$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{u}_1$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{u}_2$. By induction hypothesis, there exists terms $\mathbf{w}_1$ and $\mathbf{w}_2$ such that $\mathbf{u}_1 \longrightarrow_B^{\|} \mathbf{w}_1$, $\mathbf{v}_1 \longrightarrow_B^{\|} \mathbf{w}_1$, $\mathbf{u}_2 \longrightarrow_B^{\|} \mathbf{w}_2$, $\mathbf{v}_2 \longrightarrow_B^{\|} \mathbf{w}_2$. We take $\mathbf{w} = (\mathbf{w}_1\ \mathbf{w}_2)$.

    Or $\mathbf{v} = \mathbf{v}_3[\mathbf{v}_2/\mathbf{x}]$ with $\mathbf{t}_3 \longrightarrow_B^{\|} \mathbf{v}_3$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$, and $\mathbf{u} = ((\lambda\mathbf{x}\ \mathbf{u}_3)\ \mathbf{u}_2)$ with $\mathbf{t}_3 \longrightarrow_B^{\|} \mathbf{u}_3$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{u}_2$. Since $\mathbf{t}_2$ is a base vector, $\mathbf{u}_2$ and $\mathbf{v}_2$ are also base vectors. By induction hypothesis, there exist terms $\mathbf{w}_3$ and $\mathbf{w}_2$ such that $\mathbf{u}_3 \longrightarrow_B^{\|} \mathbf{w}_3$, $\mathbf{v}_3 \longrightarrow_B^{\|} \mathbf{w}_3$, $\mathbf{u}_2 \longrightarrow_B^{\|} \mathbf{w}_2$, $\mathbf{v}_2 \longrightarrow_B^{\|} \mathbf{w}_2$. We take $\mathbf{w} = \mathbf{w}_3[\mathbf{w}_2/\mathbf{x}]$. We have $(\lambda\mathbf{x}\,\mathbf{u}_3)\ \mathbf{u}_2 \longrightarrow_B^{\|} \mathbf{w}_3[\mathbf{w}_2/\mathbf{x}]$ by definition of $B^{\|}$. And by Proposition 16 we also have $\mathbf{v}_3[\mathbf{v}_2/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{w}_3[\mathbf{w}_2/\mathbf{x}]$.

    Or $\mathbf{v} = \mathbf{v}_3[\mathbf{v}_2/\mathbf{x}]$ with $\mathbf{t}_3 \longrightarrow_B^{\|} \mathbf{v}_3$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$, and $\mathbf{u} == \mathbf{u}_3[\mathbf{u}_2/\mathbf{x}]$ with $\mathbf{t}_3 \longrightarrow_B^{\|} \mathbf{u}_3$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{u}_2$. Since $\mathbf{t}_2$ is a base vector, $\mathbf{u}_2$ and $\mathbf{v}_2$ are base vectors also. By induction hypothesis, there exist terms $\mathbf{w}_3$ and $\mathbf{w}_2$ such that $\mathbf{u}_3 \longrightarrow_B^{\|} \mathbf{w}_3$, $\mathbf{v}_3 \longrightarrow_B^{\|} \mathbf{w}_3$, $\mathbf{u}_2 \longrightarrow_B^{\|} \mathbf{w}_2$, $\mathbf{v}_2 \longrightarrow_B^{\|} \mathbf{w}_2$. We take $\mathbf{w} = \mathbf{w}_3[\mathbf{w}_2/\mathbf{x}]$. By Proposition 16 we have both $\mathbf{u}_3[\mathbf{u}_2/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{w}_3[\mathbf{w}_2/\mathbf{x}]$ and $\mathbf{v}_3[\mathbf{v}_2/\mathbf{x}] \longrightarrow_B^{\|} \mathbf{w}_3[\mathbf{w}_2/\mathbf{x}]$.
  - Otherwise the $B^{\|}$-reduction is just an application of the congruence, i.e. $\mathbf{v} = (\mathbf{v}_1\ \mathbf{v}_2)$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$, and $\mathbf{u} = (\mathbf{u}_1\ \mathbf{u}_2)$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{u}_1$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{u}_2$. By induction hypothesis, there exists terms $\mathbf{w}_1$ and $\mathbf{w}_2$ such that $\mathbf{u}_1 \longrightarrow_B^{\|} \mathbf{w}_1$, $\mathbf{v}_1 \longrightarrow_B^{\|} \mathbf{w}_1$, $\mathbf{u}_2 \longrightarrow_B^{\|} \mathbf{w}_2$, $\mathbf{v}_2 \longrightarrow_B^{\|} \mathbf{w}_2$. We take $\mathbf{w} = (\mathbf{w}_1\ \mathbf{w}_2)$.
- If $\mathbf{t}$ is a sum then the $B^{\|}$-reduction is just an application of the congruence. The term $\mathbf{t}$ is AC-equivalent to a sum $\mathbf{t}_1 + \mathbf{t}_2$, the term $\mathbf{u}$ is AC-equivalent to a sum $\mathbf{u}_1 + \mathbf{u}_2$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{u}_1$, $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{u}_2$, and the term $\mathbf{v}$ is AC-equivalent to a sum $\mathbf{v}_1 + \mathbf{v}_2$ such that $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$ and $\mathbf{t}_2 \longrightarrow_B^{\|} \mathbf{v}_2$. By induction hypothesis, there exist terms $\mathbf{w}_1$ and $\mathbf{w}_2$ such that $\mathbf{u}_1 \longrightarrow_B^{\|} \mathbf{w}_1$, $\mathbf{v}_1 \longrightarrow_B^{\|} \mathbf{w}_1$, $\mathbf{u}_2 \longrightarrow_B^{\|} \mathbf{w}_2$, $\mathbf{v}_2 \longrightarrow_B^{\|} \mathbf{w}_2$. We take $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$.
- If finally, $\mathbf{t} = \alpha.\mathbf{t}_1$ then the $B^{\|}$-reduction is just an application of the congruence. We have $\mathbf{u} = \alpha.\mathbf{u}_1$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{u}_1$, and $\mathbf{v} = \alpha\mathbf{v}_1$ with $\mathbf{t}_1 \longrightarrow_B^{\|} \mathbf{v}_1$. By induction hypothesis, there exists a term $\mathbf{w}_1$ such that $\mathbf{u}_1 \longrightarrow_B^{\|} \mathbf{w}_1$, $\mathbf{v}_1 \longrightarrow_B^{\|} \mathbf{w}_1$. We take $\mathbf{w} = \alpha.\mathbf{w}_1$.

Proposition 18 (Hindley-Rosen lemma) *If the relations X and Y are strongly confluent and commute then the relation $X \cup Y$ is confluent.*

Theorem 1 *The system L is confluent.*

Proof. By Proposition 11, the relation $\longrightarrow_R$ is confluent, hence $\longrightarrow_R^*$ is strongly confluent. By Proposition 17, the relation $\longrightarrow_B^{\|}$ is is strongly confluent. By Proposition 15, the relations $\longrightarrow_R^*$ and $\longrightarrow_B^{\|}$ commute. Hence, by proposition 18 the relation $\longrightarrow_R^* \cup \longrightarrow_B^{\|}$ is confluent. Hence, the relation $\longrightarrow_L$ is confluent.

VI. CONCLUSION

Summary. When merging the untyped λ-calculus with linear algebra one faces two different problems. First of all simple-minded duplication of a vector is a non-linear operation ("cloning") unless it is restricted to base vectors and later extended linearly ("copying"). Second of all because we can express computable but nonetheless infinite series of vectors, hence yielding some infinities and the troublesome indefinite forms. Here again this is fixed by restricting the evaluation of these indefinite forms, this time to normal vectors. Both problems show up when looking at the confluence of the linear-algebraic λ-calculus.

The architecture of the proof of confluence seems well-suited to any non-trivial rewrite systems having both some linear algebra and some infinities as its key ingredients. The rationale is as follows:

- There is only a very limited set of techniques available for proving the confluence of non-terminating rewrite systems (mainly the notions of parallel reductions and strong confluence). Hence we must distinguish the non-terminating rules, which generate the infinities (e.g. in our case the B rule), from the others (e.g. in our case the R rules) and show that they are confluent on their own;
- The other rules being terminating, the critical pairs lemma applies. The critical pairs can be checked automatically for the non-conditional rules, but due

to the presence of infinities in the algebra it is likely that many rules will be conditional, and there is no such tool available for conditional rewriting. Hence we must distinguish the non-conditional rules (e.g. E), from the others (e.g. the F and A rules), whose corresponding pairs will have to be checked by hands;

- We then must show that the terminating rules and the non-terminating rules commute, so that their union is confluent. (e.g. in our case the conditions on B, F, A were key to obtaining the commutation, without them both subsets are confluent but not the whole.);
- If seeking to parametrize on scalars some version of the avatar's lemma is likely to be needed.

Moreover the proof of confluence entails the following, hard result, in accordance with the linearity of quantum physics:

Corollary 1 (No-cloning in the calculus) *There is no term* $\textsc{Clone}$ *such that for all term* $\mathbf{v}$*:* $(\textsc{Clone}\ \mathbf{v} \longrightarrow^* (\mathbf{v} \otimes \mathbf{v})$.

Proof. Note that the $\otimes$, **true** and **false** symbol stands the term introduced in Section IV. Say $(\textsc{Clone}\ \mathbf{v} \longrightarrow^* (\mathbf{v} \otimes \mathbf{v})$ for all $\mathbf{v}$. Let $\mathbf{v} = \alpha.\mathbf{true} + \beta.\mathbf{false}$ be in closed normal form. Then by the A-rules we have $(\textsc{Clone}\ \alpha.\mathbf{true} + \beta.\mathbf{false}) \longrightarrow^* \alpha.(\textsc{Clone}\ \mathbf{true}) + \beta.(\textsc{Clone}\ \mathbf{false})$. Next according to our supposition on $\textsc{Clone}$ this further reduces to $\alpha.(\mathbf{true} \otimes \mathbf{true}) + \beta.(\mathbf{false} \otimes \mathbf{false})$. But our supposition on $\textsc{Clone}$, also says that $(\textsc{Clone}\ \alpha.\mathbf{true} + \beta.\mathbf{false})$ reduces to $(\alpha.\mathbf{true} + \beta.\mathbf{false}) \otimes (\alpha.\mathbf{true} + \beta.\mathbf{false})$. Moreover the two cannot be reconciled into a common reduct, because they are normal. Hence our supposition would break the confluence; it cannot hold. Note that $\lambda x\ \mathbf{v}$ on the other hand can be duplicated, because it is thought as the (plans of) the classical machine for building $\mathbf{v}$ – in other words it stands for potential parallelism rather than actual parallelism. As expected there is no way to transform $\mathbf{v}$ into $\lambda x\ \mathbf{v}$ in general; confluence ensures that the calculus handles this distinction in a consistent manner.

Perspectives. The linear-algebraic λ-calculus merges higher-order computation with linear algebra in a minimalistic manner. Such a foundational approach is also taking place in [1] via some categorical formulations of quantum theory exhibiting nice composition laws and normal forms, but no explicit states, fixed point or the possibility to replicate gate descriptions yet. As for [1] although we have shown that quantum computation can be encoded in our language, the linear-algebraic λ-calculus remains some way apart from a model of quantum computation, because it allows evolutions which are not unitary. Establishing formal connections with this categorical approach does not seem an easy matter but is part of our objectives.

These connections might arise through typing. Typing is not only our next step on the list in order to enforce the unitary constraint, it is actually the principal aim and motivation for this work: we wish to extend the Curry-Howard isomorphism between proofs/propositions and programs/types to a linear-algebraic, quantum setting. Having merged higher-order computation with linear-algebra in a minimalistic manner, which does not depend on any particular type systems, grants us a complete liberty to now explore different forms of this isomorphism. For instance we may expect different type systems to have different fields of application, ranging from fine-grained entanglement-analysis for quantum computation, to opening connections with linear logic or even giving rise to some novel, quantitative logics.

We leave as an entirely open problem the search for a model of the linear-algebraic λ-calculus. One can notice already that the non-trivial models of the untyped λ-calculus are all uncountable, and hence the setting cannot be that of Hilbert spaces. This is also the reason why we have no provided a formal semantics in terms of linear operators in this paper. We suspect that models of the linear-algebraic λ-calculus will have to do with a C*-algebra endowed with some added higher-order structure – and may have a mathematical interest of its own.

Acknowledgments

Both authors would like to thank Evelyne Contejean, Jean-Pierre Jouaunnaud, Philippe Jorrand, Claude Marché and Simon Perdrix for some enlightening discussions.

[1] S. Abramsky, B. Coecke, *A categorical semantics of quantum protocols* LICS, IEEE Computer Society, 415-425, (2004).

[2] Th. Altenkirch, J. Grattage, J.K. Vizzotto, A. Sabry, An Algebra of Pure Quantum Programming, *Third International Workshop on Quantum Programming Languages*, Electronic Notes of Theoretical Computer Science, 170C, 23-47, (2007).

[3] P. Arrighi, G. Dowek, *A computational definition of the notion of vectorial space*, ENTCS **117**, 249-261, (2005).

[4] P. Arrighi, G. Dowek, *Linear-algebraic lambda-calculus*, in P. Selinger (Ed.), International workshop on quantum programming languages, Turku Centre for Computer Science General Publication, **33**, 21-38, (2004).

[5] E. Bernstein, U. Vazirani, *Quantum Complexity Theory*, Annual ACM symposium on Theory of Computing, **25**,

(1993).

[6] G. Boudol, *Lambda-calculi for (strict) parallel functions*, Information and Computation, **108**(1), 51-127, (1994).

[7] O. Bournez, M. Hoyrup, *Rewriting Logic and Probabilities*, Rewriting Techniques and Applications, LNCS **2706**, (2003).

[8] P. Boykin, T. Mor, M. Pulver, V. Roychowdhury, F. Vatan, *On universal and fault-taulerant quantum computing*, arxiv:quant-ph/9906054

[9] `http://cime.lri.fr/`

[10] N. Dershowitz, J.-P. Jouannaud, *Rewrite systems*, Handbook of theoretical computer science, Vol. **B**: formal models and semantics, MIT press, (1991).

[11] D. Deutsch, R. Josza, *Rapid solution of problems by quantum computation.* Proc. of the Roy. Soc. of London A, **439**, 553-558, (1992).

[12] D. Dougherty, *Adding Algebraic Rewriting to the Untyped Lambda Calculus*, Proc. of the Fourth International Conference on Rewriting Techniques and Applications, 1992.

[13] T. Ehrhard, L. Regnier, *The differential lambda-calculus*, Theoretical Computer Science, **309**, 1-41, (2003).

[14] A. Di Pierro, C. Hankin, H. Wiklicky, *Probabilistic λ-calculus and quantitative program analysis*, J. of Logic and Computation, **15**(2), 159-179, (2005).

[15] J.-Y. Girard. *Linear logic.* Theoretical Computer Science, **50**, 1-102, (1987).

[16] L. K. Grover, *Quantum Mechanics Helps in Searching for a Needle in a Haystack*, Phys. Rev. Lett., **79**(2), 325–328, (1997).

[17] O. M. Herescu, C. Palamidessi, *Probabilistic asynchronous pi-calculus*, ETAPS, LNCS **1784**, 146-160, (2000).

[18] J.-P. Jouannaud, H. Kirchner, *Completion of a Set of Rules Modulo a Set of Equations*, SIAM J. of Computing, **15**(14), 1155-1194.

[19] M. H. A. Newman, *On theories with a combinatorial definition of "equivalence"*, Annals of Mathematics, **43**2, 223–243, (1942).

[20] M. A. Nielsen, *Universal quantum computation using only projective measurement, quantum memory, and preparation of the 0 state*, Phys. Rev. A, **308**, 96-100, (2003).

[21] G. E. Peterson, M. E. Stickel, *Complete Sets of Reductions for Some Equational Theories*, J. ACM, **28**(2), 233-264, (1981).

[22] S. Perdrix, *State transfer instead of teleportation in measurement-based quantum computation*, Int. J. of Quantum Information , **1**(1), 219-223, (2005).

[23] R. Raussendorf, D.E. Browne, H.J. Briegel, *The one-way quantum computer - a non-network model of quantum computation*, Journal of Modern Optics, **49**, p. 1299, (2002).

[24] P. Selinger, *Towards a quantum programming language*, Math. Struc. in Computer Science, **14**(4), 527-586, (2004).

[25] P. Selinger, B. Valiron, *A lambda calculus for quantum computation with classical control*, Math. Struc. in Computer Science, **16**(3), 527-552, (2006).

[26] P.W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM J. on Computing, **26**, 1484-1509, (1997).

[27] A. Van Tonder, *A Lambda Calculus for Quantum Computation*, july 2003, arXiv:quant-ph/0307150.

[28] A. Van Tonder, *Quantum Computation, Categorical Semantics and Linear Logic*, december 2003, arXiv:quant-ph/0312174.

[29] L. Vaux, *On linear combinations of lambda-terms*, Proceedings of RTA 2007, LNCS **4533**, (2007).

[30] W. K. Wooters, W. H. Zurek, *A single quantum cannot be cloned*, Nature **299**, 802-803, (1982).